A complete class notes
Of
Computer Architecture
(BEG474CO)

B.E
Electronics & communication
VII Semester

<u>Presented by:</u>
www.oldquestions.net

Your easy access to complete study material……………………..

Syllabus:

COMPUTER ARCHITECTURE BEG474CO

Year: IV **Semester: I**

Teaching Schedule Hour/week			Examination Scheme				
Theory	Tutorial	Practical	Internal Assessment		Final		Total
2	1	3/2	Theory	Practical*	Theory**	Practical	125
			20	25	80	-	

***Continuous**
****Duration:3 hours**

Course Objective: The course provides foundation knowledge of Computer Architecture.

1. Introduction **(4 hours)**
- 1.1 History of Computer.
- 1.2 Organization and Architecture.
- 1.3 Structure and Function.
- 1.4. Pentium and Power PC Evolution.

2. Computer System. **(6 hours)**
- 2.1 Computer Components.
- 2.2 Computer Function.
- 2.3 Interconnection Structures.
- 2.4 Bus Interconnection.
- 2.5 PCI
- 2.6 Internal Memory.
- 2.7 External Memory.
- 2.8 Input/output System.
- 2.9 Operating System Support.

3. The Central Processing Unit. **(5 Hours)**
- 3.1 The architecture and Logic Unit.
- 3.2 Integer Representation.
- 3.3 Integer Arithmetic.
- 3.4 Floating-Point Representation.
- 3.5 Floating-Point Arithmetic.

4. Instruction Sets. **(6 Hours)**
- 4.1 Machines Instruction Characteristics.
- 4.2 Types of Operands.
- 4.3 Types of Operations.
- 4.4 Assembly Language.
- 4.5 Addressing.
- 4.6 Instruction Formats.

5. CPU Structure Function. **(6 Hours)**
- 5.1 Processor Organization.
- 5.2 Register Organization.
- 5.3 The instruction Cycle.
- 5.4 Instruction Pipelining.
- 5.5 The Pentium Processor .
- 5.6 The Power PC Processor.

6. Reduced Instruction Set Computers (RISC). **(7 hours)**
- 6.1 Instruction Execution Characteristics.
- 6.2 The use of Large Register File.
- 6.3 Compiler-Based Register Optimization.
- 6.4 Reduced Instruction Set Architecture.
- 6.5 RISC Pipelining.
- 6.6 The RISC versus CISC.

7. Control unit and Microprogrammed Control. **(6 hours)**
- 7.1 Micro-Operations.
- 7.2 Control of the CPU.
- 7.3 Hardwired Implementation .
- 7.4 Microinstruction Sequencing.
- 7.5 Microinstruction Execution.
- 7.6 Applications of Microprogramming.

8. Parallel Organization. **(5 hours)**
- 8.1 Multiprocessing.
- 8.2 Cache Coherence and MESI Protocol.
- 8.3 Vector Computation.
- 8.4 Parallel Processors.

Laboratory:

Student will be required to Design and Built a Project related to the computer architecture.

References:

1. Mano, Pearson Education, “ Logic and Computer Design Fundamentals”.
2. Sima, personal Education, “Advance Computer Architectures: A Design Space Approach”.
3. Heuring Pearson Education, “Computer System Design Architecture”.
4. M.Morris Mano, “Computer System Architecture”.
5. The Economics of Development and Planning by M.L.Jhingan.
6. Modern Economic Theory by K.K Dwett.

History of computer:

1. **First generation computer: (Vacuum tube):**
   - ENIAC (electronic Numerical integrator and computer) designed by and constructed under the supervision of John Mauchly and John prespereckrt at the university of pennsyevenia was the world's first general purpose electronic digital computer.

The resulting machine was enormous weighting in 30 tones, occupying 1500 sq feet floor space and containing more then 18000 vacuum tubes. When operating it consumes 140kw of power. It was substantially faster then any electro-mechanical computer being capable of 5000 addition per second.

ENIAC was decimal machine i.e numbers were represented in decimal form ring of 10 vacuum tubes represent each digit. At any time only one vacuum tube was in ON state representing one of 10 digits. The major drawback of ENIC was that it had to be programmed manually by setting swithches and plugging and unplugging cables.

In 1964, Von-Neumann and his colleagues begin the design of new stored program computer referred to as IAS computer at the Princeton institute for advance study. Figure shows the general structure of IAS computer.

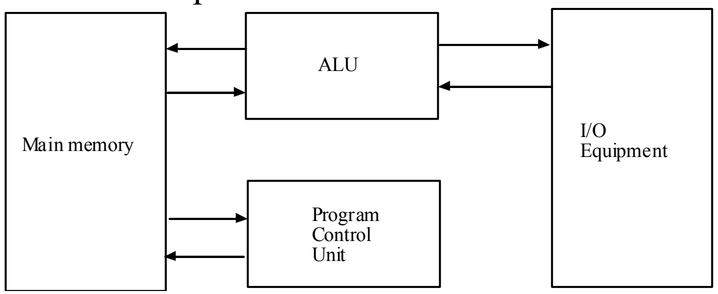


Fig. Structure of IAS computer.

It consist of following:

Main memory: which store both data and instruction(program).

ALU:ALU capable of operating on binary data.

Control unit: Which interprets the instruction and memory and causes them to be executed.

Input/ output unit: I/o equipment operated by control unit.

Second Generation:(Transistor): The first major change in electronic computer came with the replacement of vacuum tube by the transistor. The transistor is smaller cheaper and dissipates less heat then a vacuum tube but can be used in same way as a vacuum tube to construct computer. Unlike the vacuum tube which requires the wire, metal plate, glass capsule and vacuum. The transistor is sold state device made from silicon.

The use of transistor define the second generation of computer. The second generation saw the introduction of more complex arithmetic and logic unit and control unit, use of high level languages and provision of system software with the computer.

Third generation: (integrated circuit):A single self contain transistor is called discrete component. Through out the 1950's and early 1960's electronic equipment was composed largely of discrete components – transistor, capacitor, resistor and so on. Discrete component will manufacture separately packed in their own container and solder together on circuit board. Which were then instilled in computer oscilloscope and other electronic equipment. The entire manufacturing process from transistor to circuit board was expensive and cumbersome.

In 1958 came the achievement that revolutionized electronic and started the era of micro electronics; the invention of electronic circuit. It is the integrated circuit that defines third

generation of computer. The integrated circuit exploits the facts that such component as transistor resistor and conductors can be fabricated from semiconductor such as silicon. It is merely extension of solid state art to fabricate entire circuit in tiny peace of silicon rather then assemble discrete component made from separate peace of silicon. Initially only a few gates could be reliably manufacture and package together these early integrated circuit are referred as Small scale integration. (SSI).
Later generation: Beyond the third generation there is less general agreement of defining generation of computer. With the introduction of large scale integration (LSI) more then one thousand component can be placed on single integrated circuit chip define 4th generation computer. Very large scale integration VLSI achieve more then ten thousand component per chip and current VLSI chip can contain more then one lakh components per chip defines 5th generation of computer.

Date: 2065/11/8

Organization and architecture:

Computer architecture refers to those attributes of a system visible to a programmer or those attributes that help direct impact on logical execution of program. Computer organization refers to operational units and their inter connections that realize the architectural specification. Example of architectural attributes include instruction set, number of bits used to represent various data type, i/o mechanism and technique of addressing memory. Organization attributes include those hardware details transferring to the programmer such as control signal, interfaces between computer and peripheral and memory technology used.

Structure and function:

A computer is a complex system contains million of elementary electronic component.
Structure: The way in which the component are interrelated.
Function: The operation of each individual component is a part of structure.

Figure:

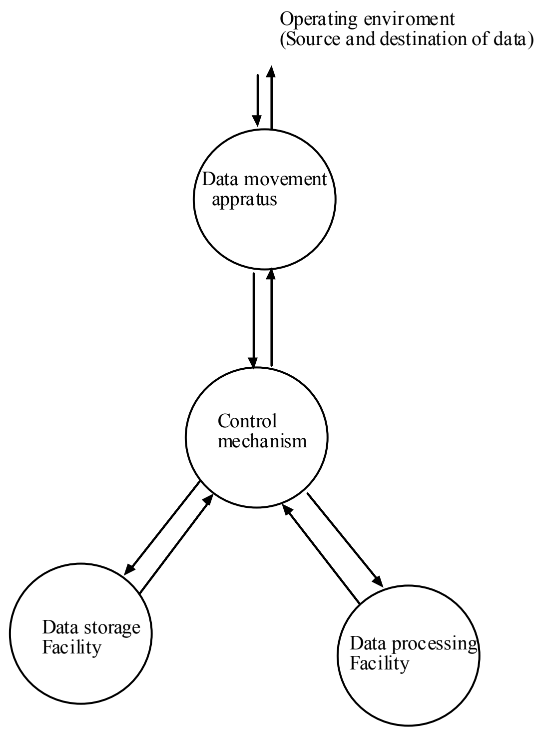


Fig. deptics the basic functions that a computer can perform. In general terms, there are only four:

- Data processing.
- Data storge.

- Data movement
- Control

Structure:

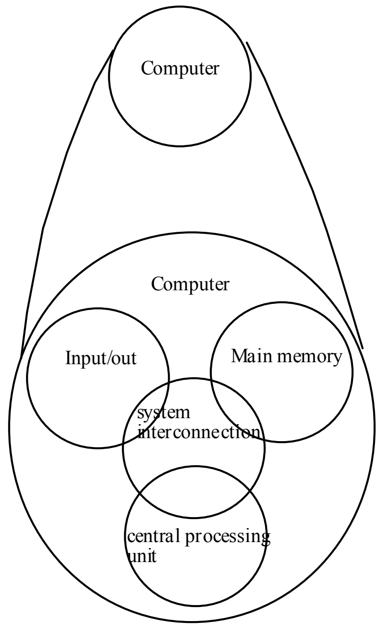


Fig: Computer: Top level structure.

There are four main structural components:

i) Central processing units: Controls the operation of computer and performs its data processing function.
ii) Main memory: Stores data.
iii) I/O : moves data between the computer and its external environment.
iv) System interconnection: Some mechanism that provides for communication among CPU, main memory and I/O.

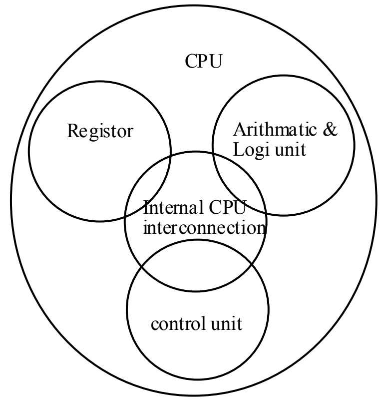


Fig: CPU

The major structural component of CPU are :
Control unit: Controls the operation of CPU
ALU: Performs the computer data processing function.
Register: provides storage internal to the CPU.
CPU interconnection: Some mechanism that provides the communication among control unit , ALU and register.

Pentium & power PC evolution:
Pentium: Some of the highlight of evolution of Intel product line.

8080: Eight bit machine with eight bit data path to the memory.
8086: 16 bit machine with wider data path and larger register and instruction queue that prefetch a few instructions before they are executed.
80286: Extension of 8086 enabled addressing 16MB memory instead of just 1 MB.
80386: 32 bit machine support multitasking meaning it could run multiple programs at the same time.

80486: Introduce the use of much more sophisticated and powerful catch technology and sophisticated and instruction pipelining.
Pentium: Pentium introduce super scalar technique which allow multiple instruction to execute in parallel.
Pentium pro: Super scalar organization with aggressive use of register renaming branch prediction.
Pentium 2: Design to process video , audio or graphics data efficiently.
Pentium3: Support 3D graphics software.
Pentium 4: Includes enhancement of multimedia.
Itanium: Makes use 64bit organization.

Power PC: The following are the principle members of power PC family.
601: 32 bit machine
603: Also 32 bit machine comparable in performance with 601. But with lower cost more efficient implementation.
604: 32 bit machine uses much more advance super scalar design technique to achieve greater performance.
620: 64 bit machine including 64 bit register and data path.
740/750: Also know as G3 processor integrates two levels of cache in the main processor chip.
G4: Increases parallelism and internal speed of processor chip.

Date:2065/11/13

Chapter- 2
COMPUTER SYSTEM:
Computer components:

Figure:

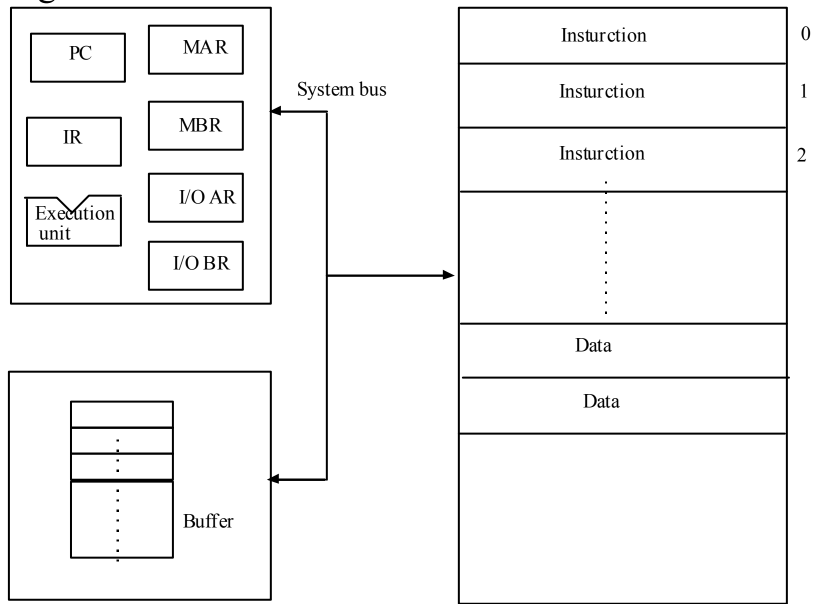


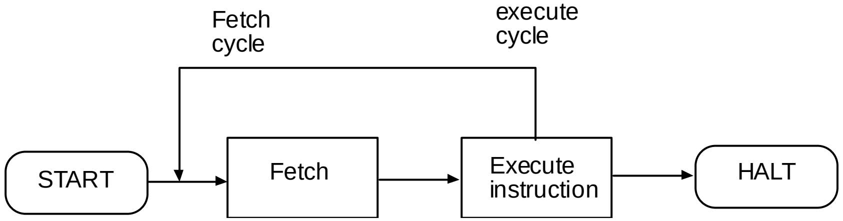
Fig:- Computer components.

PC – program counter
IR – Instruction register.
MAR- memory address register.
MBR- memory buffer register.
I/O AR – input/output address register.
I/O BR – Input/output buffer register.

The central processing unit (CPU) exchanges data with memory for this purpose it typically makes used of two internal (to the cpu) register MAR which specify the address in memory for next R/W and MBR which contents the data to be written into the memory or received the data from the memory similarly I/O AR specify the particular i/o device. I/O BR register is used for exchange of data between I/O module and CPU.

A memory module consists of set of location defined by sequential number address each location content a binary no that can be interpreted as a instruction or data. I/O module transfer the data form external device to CPU and memory vice-versa. It contains internal buffer for temporarily holding these data until they can be sent on.

Computer function: The basic function performed by a computer is execution of program which consist of set of instruction stored in memory. Instruction processing consists of two steps:
processor reds (fetches) instruction from memory one at a time and executes each instruction.
The processing requires for single instruction is called instruction cycle.
Figure shows basic instruction cycle:



At the beginning of each instruction cycle the processor fetches the instruction from a memory. Program counter holds the address to be fetched next. Unless told other wise the processor always increment programmer counter after each instruction phase so that it will fetch next instruction in sequence.
The fetched instruction is loaded into instruction register. The instruction contains bits that specifies the action the processor is to take. The processor interprets the instruction an performs the required action. In general this actions fall into four category.
Processor memory: Data may be transferred from processor to memory or memory to processor

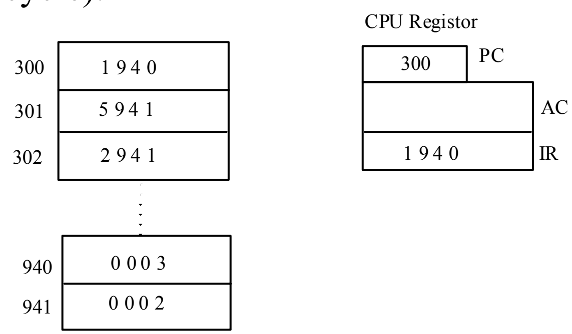
.Processor I/O: Data may be transferred to or form peripheral device by transferring between processor and I/O memory.

Data processing: The processor may perform some arithmetic or logic operation on data.

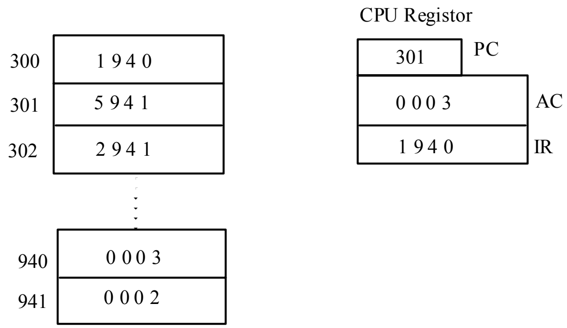
Control: An instruction may specifies that the sequence of execution be alter.

Date: 2065/11/14

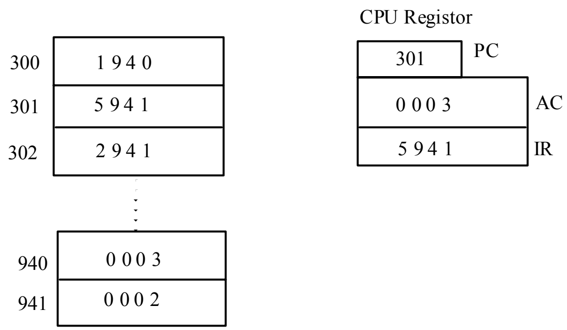
Computer Function:
Step:1 (fetch cycle):



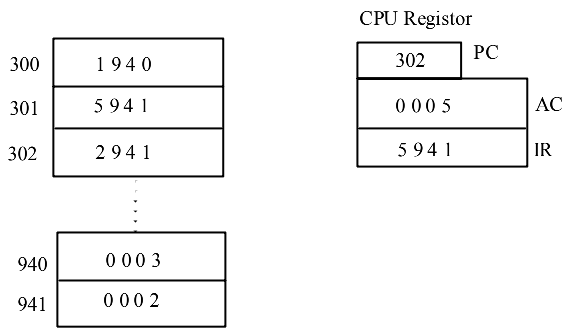
Step:2



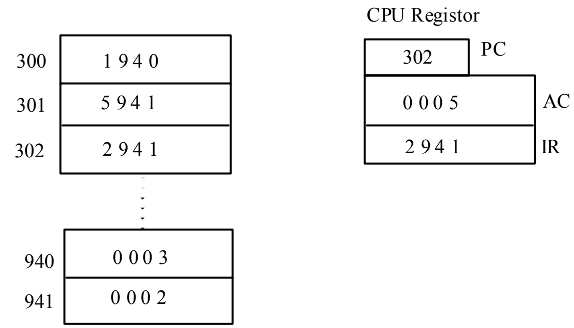
Step: 3



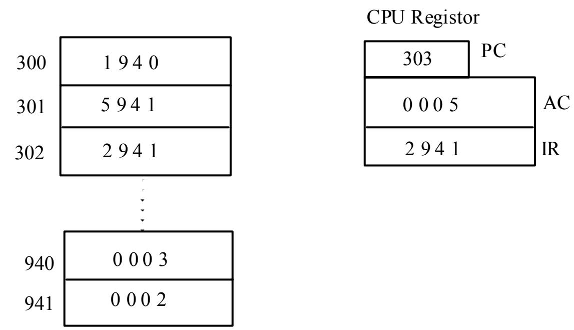
Step: 4



Step:5



Step: 6



The program fragment shown adds the contents of memory words at address 940 to the contents of memory word at address 940 and stores the result in latter location.

Three instruction which can be describe as three fetch and three execute cycles are require:

1. Pc contains 300, the address of 1^{st} instruction. This instruction is loaded into the IR and PC is incremented.
2. The first four bits in IR indicate that AC is to be loaded . The remaining 12 bits specify the address (940) from which data are to be stored.
3. The next instruction 5941 is fetch from the location 301 and PC is incremented.

4. The old contents of AC , and contents of location 941 are added and the result is stored in AC.
5. The next instruction 2941 is fetch from location 302 and PC is incremented.
6. The contents of AC are stored in 941.

To accommodate interrupt, an interrupt cycle is added to the instruction cycle as shown in fig.

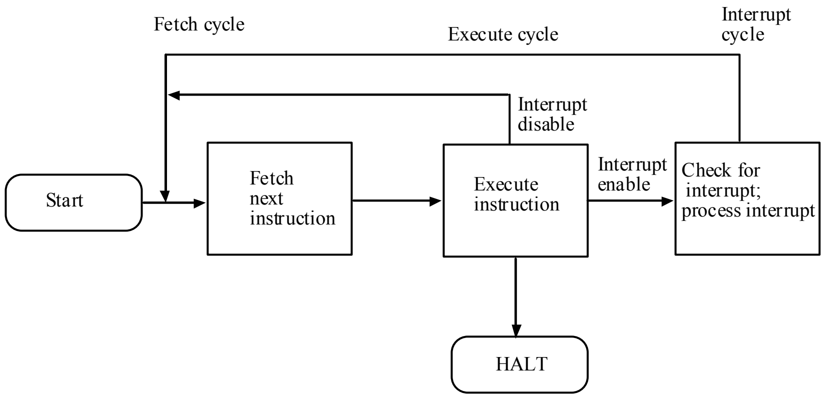
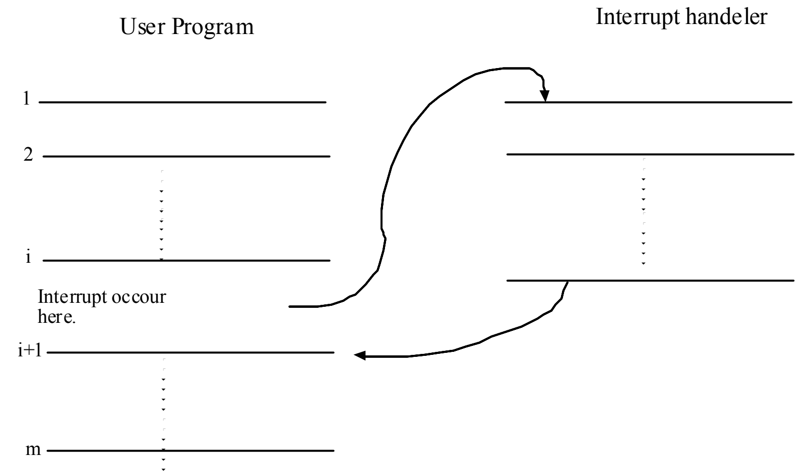


Fig: Instruction cycle with interrupt.

In the interrupt cycle the processor checks to see if any interrupt have occur , indicated by the presence of interrupt signal. If no interrupt are pending , the processor proceeds to fetch cycle and fetch the next instruction of current program of interrupt is pending, the processor does the following:

1. It suspense the execution of current program being executed and saves its content.
2. It sets the program counter to starting address of interrupt and routine.



Interconnection Structure:

A computer consist of set of components or module of three basic types, (processor , memory , I/O) that communicate with each other. The collection of path connecting various module is called interconnection structure. The design of this structure will depend on exchanges that must be made.

Figure suggest the type of exchanges that must be needed by indicating the major for of input and output for each module type.

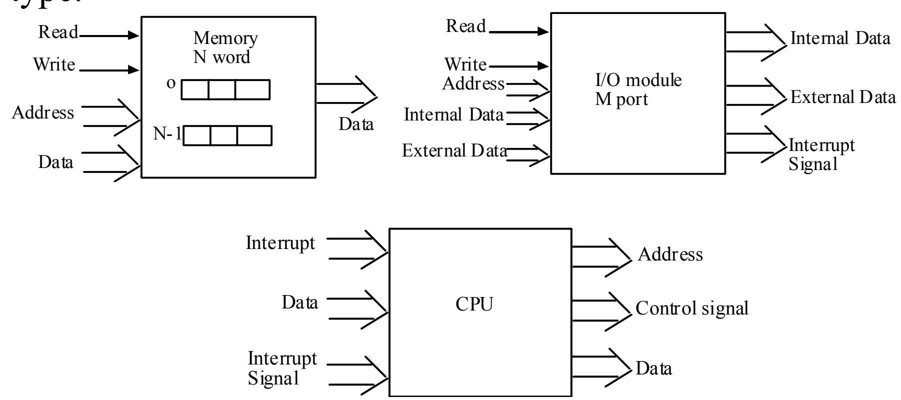


Fig: Computer module

Date: 2065/11/15

The interconnection structure must support the following type of transfer :

1. Memory to processor
2. Processor to memory
3. I/O to processor
4. Processor to I/O
5. I/O to or from memory.

Bus interconnection: A bus is communication path way connecting two or more devices. A key characteristics of bus is that it is a share transmission medium. Typically a bus consists of multiple communication path ways or lines. Each line is capable of transmitting signal representing binary 1 and binary 0. Several line of bus can be used to transmit binary digit simultaneously (in parallel). For example 8 bit unit of data can be transmitted over 8 bus lines. Computer system contains a number of different buses that provide path ways between components at various level of computer system hierarchy. A bus that connect major computer components (processor, memory, I/O) is called system bus. The lines can be classified into three functional groups data, address and control lines.

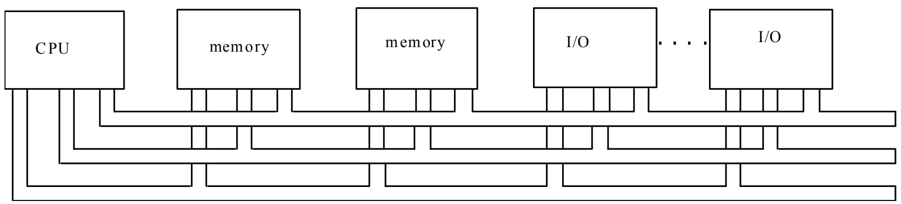


Figure: Bus interconnection Scheme.

Physically the system bus is actually a number of parallel electrical conductors in the classic bus arrangement these conductors are metal lines etched in board as shown in figure.

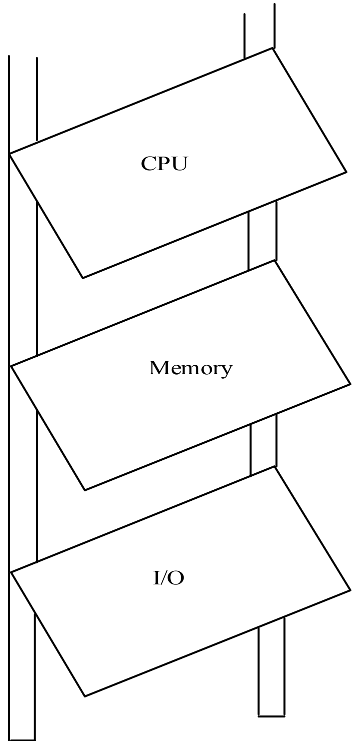
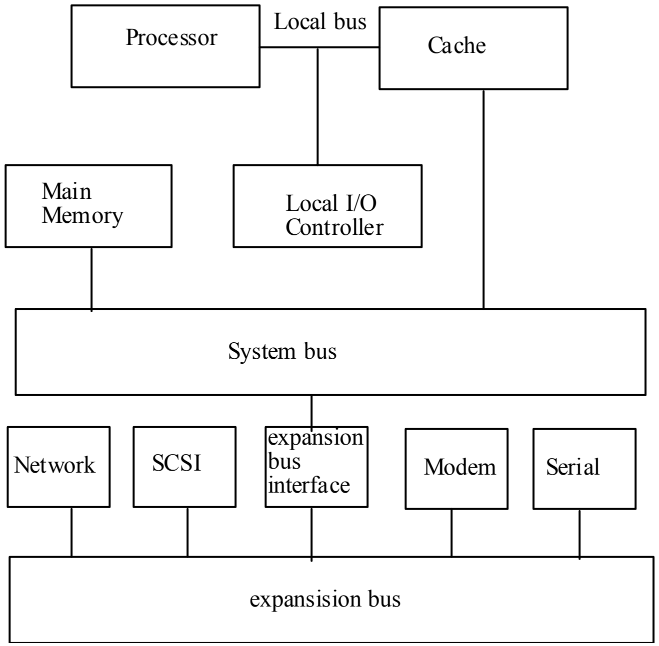


Fig: Typical physical realization of bus architecture.

If a great number of devices are connected to the bus performance will suffer. In general the more devices attached to the bus length and hence the greater propagation delay. Most computer system used multiple buses. A typical traditional structure is shown in figure.



SCSI= small computer system interface

Fig. Traditional bus architecture.

The use of cache structure insulates the processor from requirement to access main memory frequently. I/O transfers to and from main memory across the system bus do not interfere with the processors activity. An expansion bus interface buffers data transfer between the system bus and I/O controllers. These tradition bus architecture is reasonably efficient but begins to breakdown as higher and higher performance is seen in the I/O devices. In response to these growing demands common approach taken by industry is to built high speed bus that is closely integrated with rest of the system requiring only bridge between the processors bus and high speed bus.

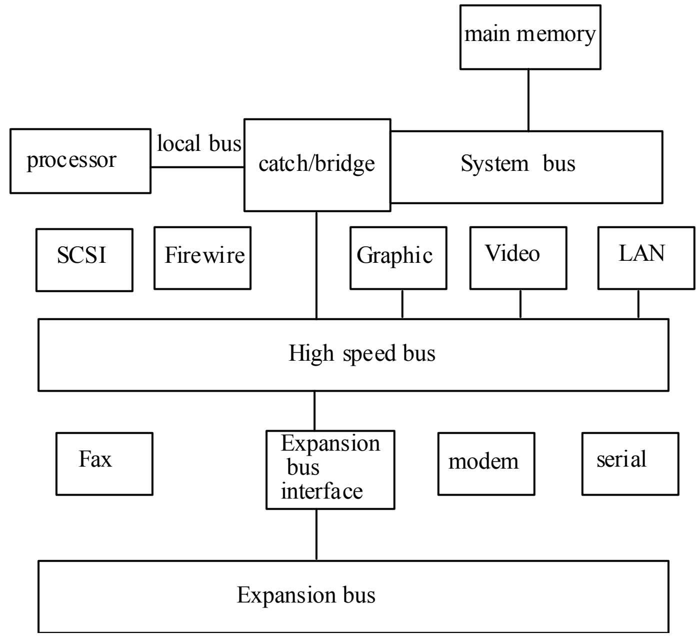


Figure: High performance architecture (Mezzanine architecture)

Date:2065/11/19

PCI(Peripheral Component interconnection):

Peripheral component interconnect is a popular high bandwidth processor independent bus that can function as peripheral bus compared with other common bus specification. PCI delivers better system performance for high speed I/O Sub system(network interface controller). PCI is design to support a variety of microprocessor base configuration including both single and multiple processor system. Fig shows typical use of PCI in single processor system.

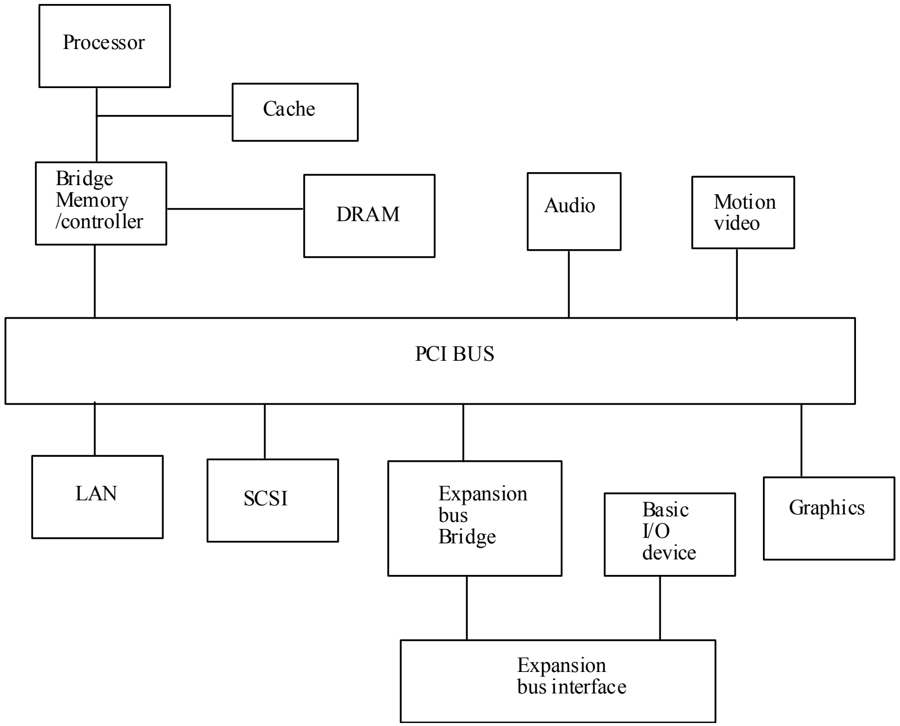


Fig: Typical desktop system.

A combine DRAM controller and bridge a PCI bus provides tight coupling with the processor and ability to deliver data at high speed. The bridge acts data buffer so that the speed of PCI bus may differ from that of processor I/O capability.

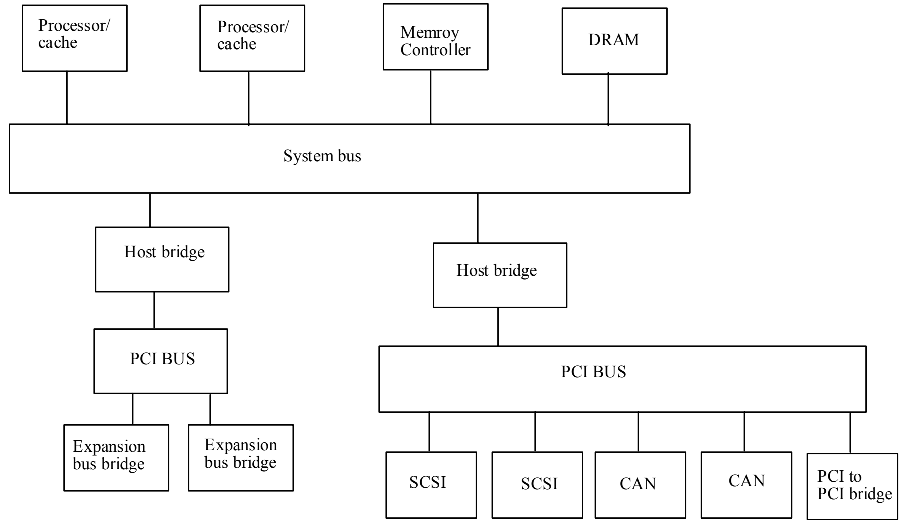


Fig: Typical server system.

In multiprocessor system one or more PCI configuration may be connected by bridges to processor system bus. The system bus supports only the processor/catch unit, main memory and PCI bridge.

Internal memory:

RAM: One distinguishing characteristics of RAM is that it is possible both to read data from the memory and write new data into the memory easily and rapidly. Other distinguishing characteristics of RAM is that it is volatile. Ram most be provided with constant power supply. The two tradition form of RAM used in computer are DRAM and SRAM.

DRAM: DRAM is made with a cells that stores data as charge on capacitor. The presence or absence of charge on capacitor is interpreted as binary 1 or 0. Because capacitor have a natural tendency to discharge , DRAM requires periodic charge refreshing to maintain data storage.

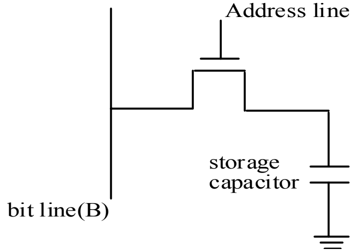


Fig: DRAM cell

The address lines is activated when the bit value from the cell is to be read or written. The transistor acts as switch.

For write operation voltage signal is applied to the bit line, a high voltage represents 1 and low voltage represents 0. A signal is then applied to the address line allowing charge to be transferred to the capacitor. For read operation when address line selected the transistor turn ON and charge stored on capacitor is fed out on to bit line.

SRAM:

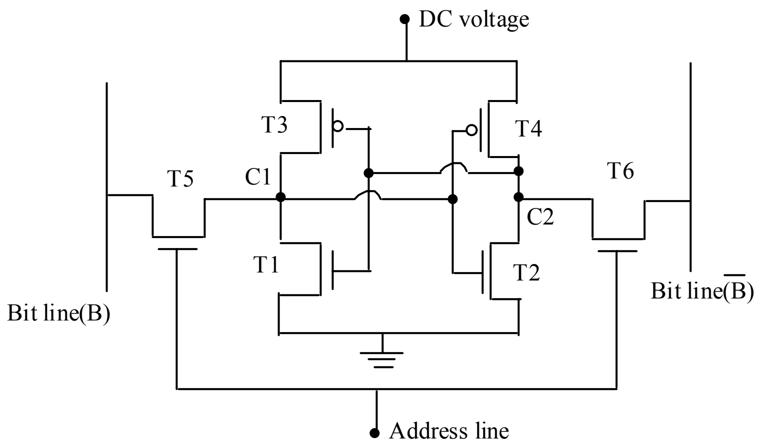


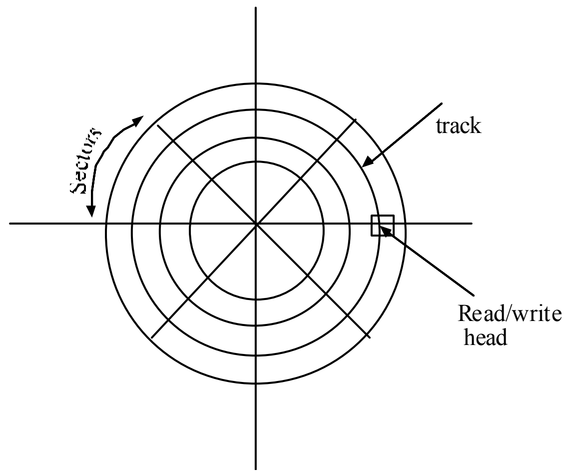
Figure: SRAM line.

Four transistor T_1, T_2 , T_3 , T_4 are cross connected in arrangement that produce a stable logical state. In logic state 1 pint c_1 and high and point c_2 is low. In this state T_1 and T_4 are off and T_3 and T_2 and on. As in the DRAM the address line is used to open or close a switch. The address lines control two transistor T_5 and T_6. When a signal is applied to this line the two transistor are switch on allowing read or write operation.

Date:2065/11/22

External memory (Auxiliary memory): To understand fully the physical mechanism of external memory devices one must have a knowledge of magnetic electronics and electromechanical systems. Although the physical properties of there storage devices can be quite complex. Their logical properties can be characterized by few parameters. The important characteristics of any devices are its access mode, access time, transfer rate capacity and cost.

Magnetic disk:



A magnetic disk is a circular plate constructed with metal or plastic coated with magnetic material often both side of disk are used and several disk stacked on one spindle which Read/write head available on each surface. All disk rotate together at high speed. Bits are stored in magnetize surface in spots along concentric circles called tracks. The tracks are commonly divided into sections called sectors. After the read/write head are positioned in specified track the system has to wait until the rotating disk reaches the specified sector under read/write head. Information transfer is very fast once the beginning of sector has been reached.

Disk that are permanently attached to the unit assembly and can not be used by occasional user are called hard disk drive with removal disk is called floppy disk.

RAID(Redundant Array Independent Disk):

Disk storage designers recognize that if one component can only be pushed so far addition gain in the performance are to be had by using multiple parallel components in the case of disk storage this leads to the development of arrays of disk that operate independently and in parallel with multiple disk separate I/O request can be handled in parallel as long as the data reside on separate disk. Further single I/O request can be executed in parallel if the block of data to be accessed is distributed across multiple disk.

With the use of multiple disk there is wide variety of ways in which data can be organized and in which redundancy can be added to improve reliability. This could make it difficult to develop data base scheme that are usable on number of plat form and operating system. Fortunately industry has agreed on standardized on scheme for multiple disk data base design know as RAID.

Optical memory: The huge commercial success of CD enabled the development of low cost optical disk storage technology that has revolutionized computer data storage. The disk is form from resin such as polycarbonate. Digitally recorded information is imprinted as series of microscopic pits on the surface of poly carbonate . This is done with the finely focused high intensity leaser. The pited surface is then coated with reflecting surface usually aluminum or gold. The shiny surface is protected against dust and scratches by the top coat of acrylic.

Information is retrieved from CD by low power laser. The intensity of reflected light of laser changes as it encounter a pit. Specifically if the laser beam falls on pit which has some what rough surface the light scatters and low intensity is reflected back to the surface. The areas between pits are called lands. A land is a smooth surface which reflect back at higher intensity. The change between pits and land is detected by photo sensor and converted into digital signal. The sensor test the surface at regular interval.

Magnetic tape: Tape system used the same reading and recording technique as disk system. The medium is flexible polyester tape coated with magnetizable material.

Data on tapes are structured as number of parallel tracks running length wise. Earlier tape system typically used nine tracks. This made it possible to store data one byte at a time with additional parity bit as 9^{th} track. The recording of data in this form is referred to as parallel recording.

Date:2065/11/29

Input/output system:

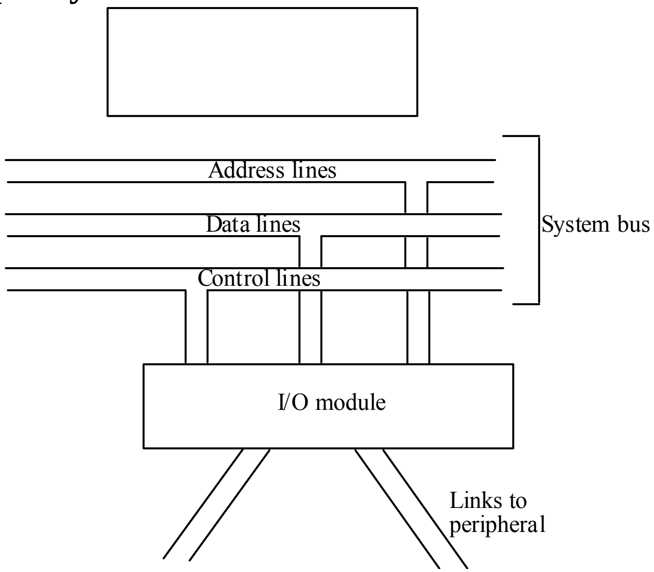


Figure: Model of I/O module

The computer systems I/O architecture is its interface to the outside world. An external device attached to the computer by a link to an I/O module. The link is used to exchange control, status and data between the I/O port and external device. An external device connected to I/O module is often referred to as peripheral device or simply peripheral.

We can broadly classify external device into 3 categories.

1) Human readable; suitable for communicating with computer user.
2) Machine readable; Suitable for communicating with equipment.
3) Communication: Suitable for communicating with remote devices.

Examples of human readable devices are VDV and printers. Examples of machine readable devices are magnetic discs and tapes. Communication devices allow a computer to exchange data with remote device. Which may be a human readable device, a machine readable device or another computer.

The most common means of computer/user interaction is keyboard/monitor arrangement. The user provides input through the keyboard. This input is then transmitted to the computer and may also be displayed on monitor. In addition, the monitor display the data provided by the computer.

In very general terms, the nature of external devices is indicated in fig below.

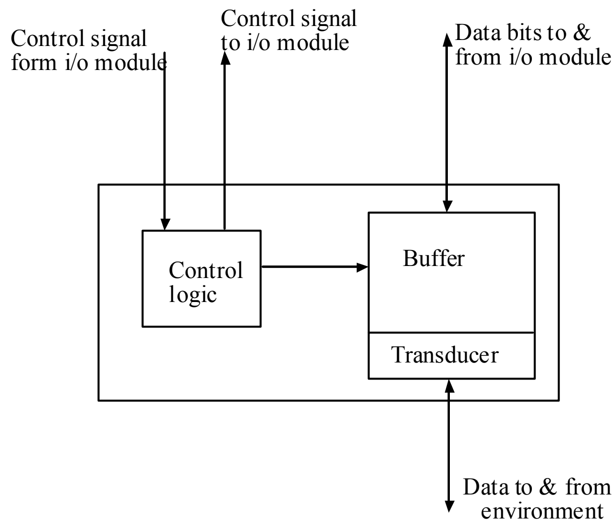


Fig: Block diagram of external device.

Operating system Support:

Operating system is a program that controls the execution of application program and acts as interface between the user of computer and computer hardware. The os as a user computer interface. The hardware and software used in providing application to a user can be viewed in hierarchical fashion as depicted in fig.

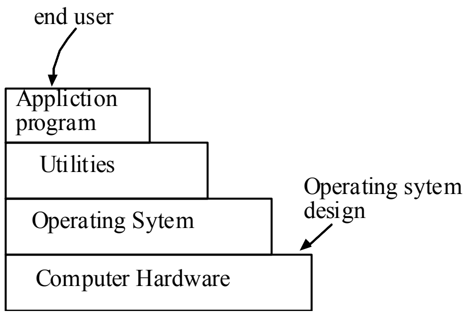


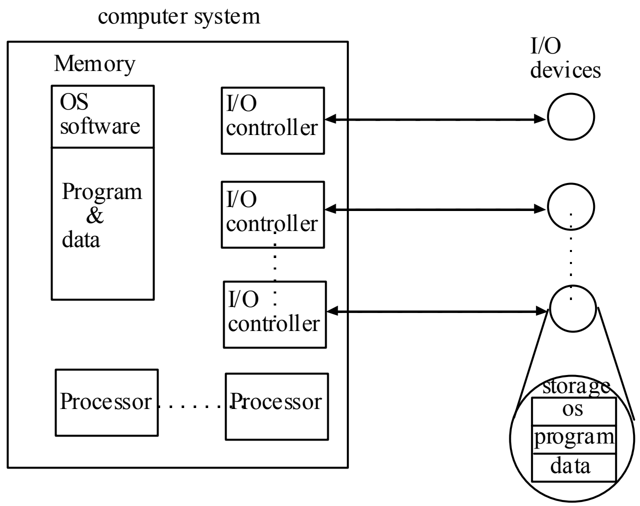
Fig: layers and view of computer system

The end user use a computer system in terms of application that application can be expressed in programming language and is developed by application program. If one were to develop application program as set of processor instruction that is completely responsible for controlling computer hardware, one would face with complex task. To each this task a set of system program is provided. Some of these program are referred to as utilities. These implement frequently used function that assist in program creating, management of file and control of I/o devices. The operating system acts mediator making it easier for programmer to access and use those facilities and services.

Briefly the operating typically provides services in the following areas.:

1. Program creation: The operating system provides a variety of facility and services such as editors and debuggers to assist the programmers in creating program.
2. Program execution: A number of task need to be performed to execute a program. Instruction and data must be loaded into main memory, I/O devices and file must be initialized and other resources must be prepared. The operating system handles all o f this for the user.
3. Access to I/O devices: Each i/o devise requires its own peculiar set of instruction or control signal for operation. The operating system takes care of this details so that program can think in terms of simple read and write.
4. Controlled access to file: In this case of system with multiple simultaneous user, the operating system can provide protection mechanism to control access to the file.
5. System access: The access function must provide protection of resources and data from unauthorized users.
6. Errors detection & response: A variety of errors can occur while compute system is running. These include internal and external hardware errors. Such as memory errors, device faller or various software errors such as arithmetic over flow. In each case operating system must make the response that clears the error condition.
7. Accounting: A good operating system will collect usages statistics for various resources and monitor performance parameter such as response time.

The Operating system as Resource manger:



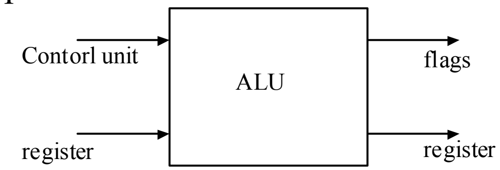
A computer is a set of resources for the movement storage and processing of data for the control of these function. The operation system is responsible for managing these resources. Figure suggest the main resources that are managed by operating system.

A portion of operating system is in main memory. The reminder of main memory contains other user programs and data. The operating system decide when i/o device can be used by a program in execution and controlled access to and use of files. The processor is itself resources and the os must determine how much processor time is to be devoted to the execution of particular user program.

Date: 2065/12/4

Arithmetic and Logic Unit : ALU is the part of computer that actually performs arithmetic and logical operations on data. All of the other elements of computer system- control unit, registers, memory, I/O are their mainly to bring data into the ALU for it to process and then to take the result back out.

An ALU & indeed all electronic components in computer are based on the use of simple digital logic device that can store binary digit and perform simple Boolean logic function. Figure indicates in general in general term how ALU is interconnected with rest of the processor.



Data are presented to ALU in register and the result of operation are stored in register. These registers are temporarily storage location within the processor that are connected by signal path to the ALU. The ALU may also set flags as the result of an operation. The flags values are also stored in registers within the processor. The control unit provide signals that control the operation of ALU and the movement of data into an out of ALU.

Integer Representation: (Fixed-point representation):

An eight bit word can be represent the numbers form zero to 255 including

00000000 = 0
00000001 = 1

11111111 = 255

In general if an n-bit sequence of binary digits a_{n-1}, a_{n-2} …..a_1, a_o Is interpreted as unsigned integer A. It's value is

$A = \sum^{n-1}_{i=0} 2^i a_i$

Sign magnitude representation:

There are several alternative convention used to represent –ve as well as +ve integers, all of which involves treating the most significant (left most) bit in the word as sign bit. If the sign bit is 0 the number is +ve and if the sign bit is 1 the number is –Ve. In n bit word the right most n-1 bits hold the magnitude of integer.

E g.

+18 = 00010010

- 18 = 10010010 (sign magnitude)

The general case can be expressed as follows:

$A = \sum^{n-2}_{i=0} 2^i a_i$ if $a_{n-1} = 0$.

$= -\sum^{n-2}_{i=0} 2_i a_i$ if $a_{n-1} = 1$

There are several drawbacks to sign-magnitude representation. One is that addition or subtraction require consideration of both signs of number and their relative magnitude to carry out the required operation. Another draw back is that there are two representation of zero. Eg.

$+0_{10}$ = 00000000

-0_{10} = 10000000 which is inconvenient.

Date:2065/12/5

Twos complement representation:

Like sign magnitude tows complement representation uses the most significant bit as sign bit making it easy to test weather the integer is negative or positive. Differs from the use of sing magnitude representation in the way that other bits are interpreted. For negation take the Boolean complement of each bit of corresponding positive number, then add one to the rustling bit pattern viewed as unsigned integer.

Consider n bit integer A in twos complement representation. If A is +ve then the sign bit a_{n-1} is zero. The remaining bit represent the magnitude of the number.

$A = \sum^{n-2}_{i=0} 2^i a_i$ for $A \geq 0$

The number zero is identified as +ve and therefore has zero sign bit and magnitude of all 0's. We can see that the range of +ve integer that may be represented is from 0 (all the magnitude bits are zero) through 2^{n-1}-1 (all of the magnitude bits are 1.)

Now for –ve number integer A. The sign bit a_{n-1} is 1. The range of –ve integer that can be represented its from -1 to -2^{n-1}

Twos complement, $A = -2^{n-1} a_{n-1} + \sum^{n-2}_{i=0} 2^i a_i$

Defines the twos complement of representation of both positive and negative number.

E.g

Decimal	Sign magnitude representation	Twos complement representation
+7	0111	0111
-7	1111	1001

-128	64	32	16	8	4	2	1

(a) Eight-position twos complement value box.

-128	64	32	16	8	4	2	1
1	0	0	0	0	0	1	1

-128 +2 +1 = -125

(b) Convert 10000011 to decimal

-128	64	32	16	8	4	2	1
1				1			

-120 =-128 +8

(c) Convert decimal -120 to binary

Fig. use of value box for conversion between 2's complement binary and decimal.

Converting between different bit lengths:

It is some time desirable to take n bit integer and store it in m bit where m greater then n. In sign magnitude notation this easily accomplished: simply move the sign bit to the new left most position and fill in with zero.

+18= 00010010 (sign magnitude , 8 bits)

+18= 0000000000010010(sign magnitude 16 bit)

-18= 10010010 (sign magnitude , 8 bit)

-18= 1000000000010010(sign magnitude , 16bit)

This procedure will not work for 2's complement –ve integer.

-18= 11101110 (2's complement, 8 bits)

-32,658 = 1000000001101110 (2's complement , 16 bits)

Instead the rules for 2's complement integer is to move the sign bit to the new left most position and fill in with copies of sign bit. For +ve numbers fill in with zero and for –ve numbers fill in with 1's. This is called sign extension.

-18= 11101110 (2's comlemetn , 8 bit)

-18= 1111111111101110

To see why this rule work, let us again consider n bit sequence of binary digits. $a_{n-1}a_{n-2}\ldots\ldots a_1a_o$ interpreted as twos complement integer so that its value is $A = -2^{n-1}a_{n-1}+\sum^{n-2}_{i=0} 2^i a_i$

If A is +ve number the rule clearly works , now if A is –ve we want to construct m bit representation with n>m.

$A = -2^{m-1}a_{m-1}+\sum^{m-2}_{i=0} 2^i a_i$

The two values must be equal,

$-2^{m-1}a_{m-1}+\sum^{m-2}_{i=0} 2^i a_i = -2^{n-1}a_{n-1}+\sum^{n-2}_{i=0} 2^i a_i$

$-2^{m-1}+\sum^{m-2}_{i=0} 2^i a_i = -2^{n-1}+\sum^{n-2}_{i=0} 2^i a_i$

$2^{n-1}+\sum^{m-2}_{i=n-1} 2^i a_i = 2^{m-1}$

$1+\sum^{n-2}_{i=0} 2^i +\sum^{m-2}_{i=n-1} 2^i a_i =1+\sum^{m-2}_{i=0} 2^i$

$\sum^{m-2}_{i=n-1} 2^i a_i= \sum^{m-2}_{i=n-1} 2^i$

i.e

$a_{m-2} = \ldots\ldots = a_{n-1} = 1$

Date: 2065/12/6

Integer arithmetic:

Negation: N bit a sequence of binary digit $a_{n-1}a_{n-2}\ldots a_1a_o$ as twos complement integer A. So that its value,

$A = -2^{n-1}a_{n-1}+ \sum^{n-2}_{i=0}2^i a_i$

Now form the bit wise complement a_{n-1}(comp)a_{n-2}(comp)….a_1(comp)a_o(comp) and treating this unsine integer and add 1. Finally interpreter the result in n bit sequence of binary digit as tows complement integer B. So that its value is

$B = -2^{n-1}a_{n-1}(comp)+\sum^{n-2}_{i=0}2^i a_i(comp)$

Now we want, A = -B which means A+B=0

$A+B = -2^{n-1}a_{n-1}+ \sum^{n-2}_{i=0}2^i a_i-2^{n-1}a_{n-1}(comp)+\sum^{n-2}_{i=0}2^i a_i(comp)+1$

$= -2^{n-1}(a_{n-1}+a_{n-1}(comp))+\sum^{n-2}_{i=0}2^i(a_i+a_i(comp))+1$

$= -2^{n-1}+\sum^{n-2}_{i=0}2^i = -2^{n-1}+1+2^{n-1}-1 = 0$

Addition and Substraction:

```
1001 = -7
0101 = +5
---------
1110 =-2
```
(a) (-7)+(+5)

```
 0011 = 3
 0100= 4
---------
 0111= 7
```
(c) (+3)+(+4)

```
0101 =5
0100 =4
---------
1001=overflow
```
(e) (+5)+(+4)

```
1100 = -4
0100 = +4
---------
10000 = 0
```
(b) (-4)+(4)

```
1100 = -4
1111 = -1
---------
11011 = -5
```
(d) (-4)+(-1)

```
1001 = -7
1010 = -6
---------
10011 = overflow
```
(f) (-7)+(-6)

The first four examples illustrate successful operation if the result of the operation is +ve then we get +ve number in ordinary binary notation. If the result of the operation is –ve we get negative number in twos complement form. Note that in some instants there is carry bit beyond the end of what which is ignore.

On any addition the result may larger then can be held in word size being use. This condition is called over flow. When overflow occur ALU must signal this fact so that no attempt is made to use the result. To detect overflow the following rule observed. If two number are added, and they are both +ve or both –ve. Then overflow occurs if and only if the result has the opposite sign.

The figure suggest the data path and hardware elements need ot accomplish addition and subtraction.

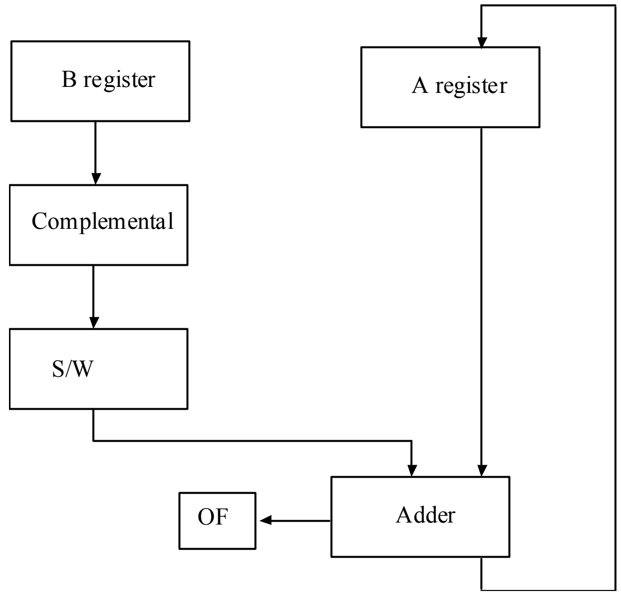


Fig: Block diagram of hardware for subtraction and addition.

```
    1011     Multiplicand 11
    1101     Multiplier 13
   ------
    1011  ┐
   0000   │  partial product
  1011    │
 1011     ┘
 ----------
 10001111    product (143)
```

Fig. Multiplication of unsigned binary integers.

1. The multiplication involve the generation of partial product 1 for each digit in the multiplier. This partial products are then sum to produce final product.
2. The partial product are easily define. when the multiplier bit is zero the partial product is zero. When the multiplier is 1 the partial product is the multiplicand.

3. The total product is produce by summing the partial products. For this operation each successive partial product is shifted one position to the left relative the perceiving partial product.
4. The multiplication of two n bit binary integer results in product of upto 2n bits in length. Eg. 11 ×11= 1001

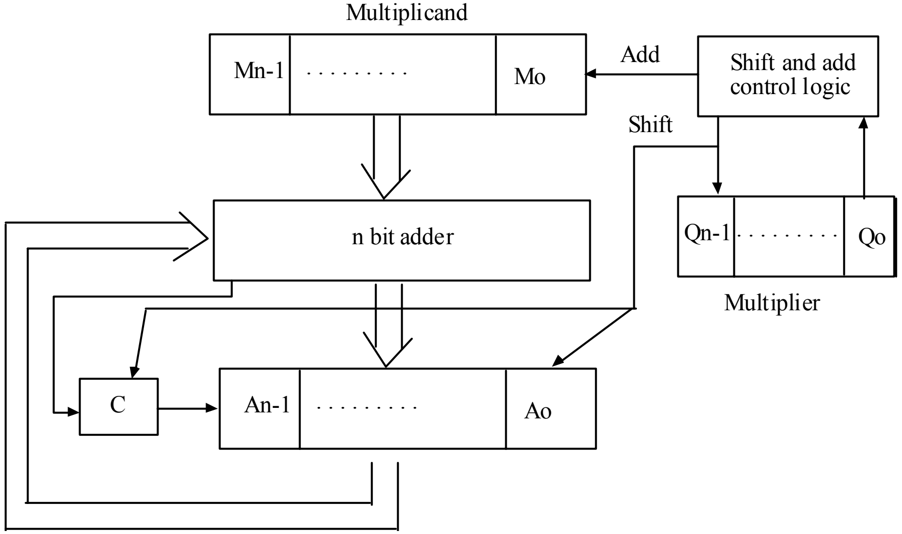


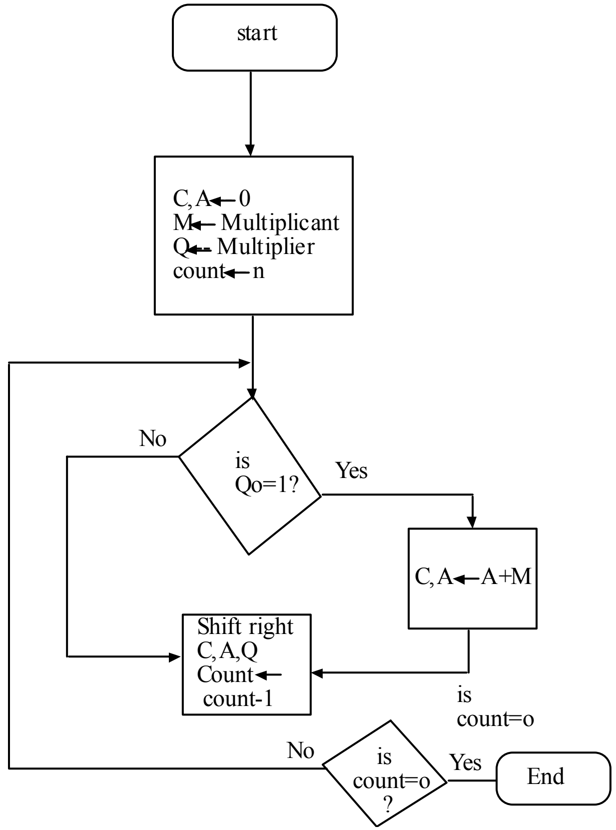
Fig: (a) block diagram.

C	A	Q	M	
0	0000	1101	1011	Initial values
0	1011	1101	1011	Add
0	0101	1110	1011	Shift
0	0010	1111	1011	shift
0	1101	1111	1011	add
0	0110	1111	1011	shift
1	0001	1111	1011	Add
0	1000	1111	1011	Shift

(b) examples from fig (i) (product in A,Q)

Date: 2065/12/18

Control logic reads bits of multiplier one at a time. If Q0 is 1 the multiplicand is added to A register and result is stored in A register with C bit used for overflow then all of the bits of C , A , and Q register are shifted to the right one bit so that C bit goes into A_{n-1}, A_0 goes into Q_{n-1} and Q_0 is lost. If Q_0 is zero and no addition is perform , just the shift. This process is repeated for each bit of the original multiplier. The resulting 2n bit product is contain in A and Q register. A flow chart of the operation is shown in fig.



2's complement multiplication: If we multiply 11 (1011) by 13 (1101) we get 143 (10001111). If we interpret this as two's

complement numbers we have, - 5 (1011) times -3 (1101) equals -113(10001111).This example illustrate that straight forward multiplication will not work if both the multiplicand and multiplier are negative. In fact it will not work if either the multiplicand or multiplier is negative. The problem is that each contribution of negative multiplicand as a partial product must be negative on 2n bit field. The sign bit of partial product must line up.

```
  1001 (9)
  0011 (3)
00001001
00010010
--------
00011011(27)
```
(a) unsigned integer.

```
  1001 (-7)
  0011 (3)
11111001
11110010
--------
11101011 (-27)
```
(b) 2's complement integer.

Fig: comparison of multiplication of unsigned and twos complement integer.

Booth's algorithm: It has the benefit of speeding of multiplication process relative to more straight forward approach. Both algorithm is depicted in figure.

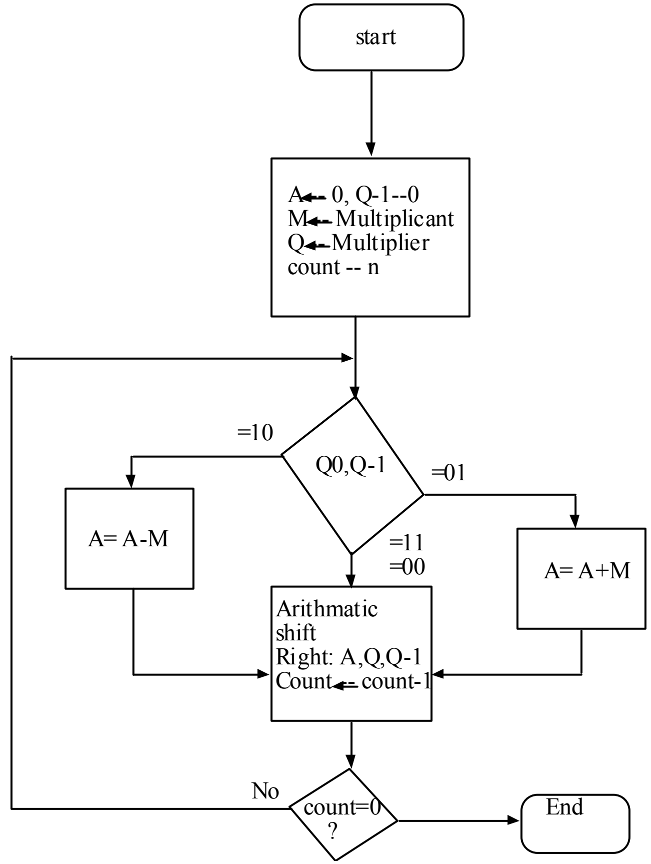


Fig. Booth's algorithm for 2's complement multiplication.

Date: 2065/12/19

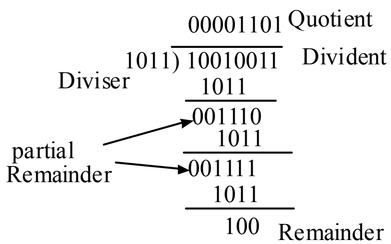
A	Q	Q_{-1}	M	
0000	0011	0	0111	Initial values.
1001	0011	0	0111	A←A-M
1100	1001	1	0111	Shift
1110	0100	1	0111	Shift
0101	0100	1	0111	A←A+M
0010	1010	0	0111	shift

0001 0101 0 0111 Shift

Fig. Examples of Booth's algorithm (7 x 3)

Multiplier and multiplicand are placed in Q and M register respectively. There is also one bit register placed logically to the right of the least significant bit Q_o of the Q register and designated as Q_{-1}. The result of multiplication will appear in A and Q resister. A and Q_{-1} are initialized to zero if two bits (Q_o and Q_{-1}) are the same (1 – 1 or 0 -0) then all the bits of A , Q and Q_{-1} registers are shifted to the right 1 bit. If the two bits differ then the multiplicant is added to or subtracted from the A register depending on weather the two bits are 0-1 or 1-0 . Following the addition or subtraction the right shift occurs.

Division:



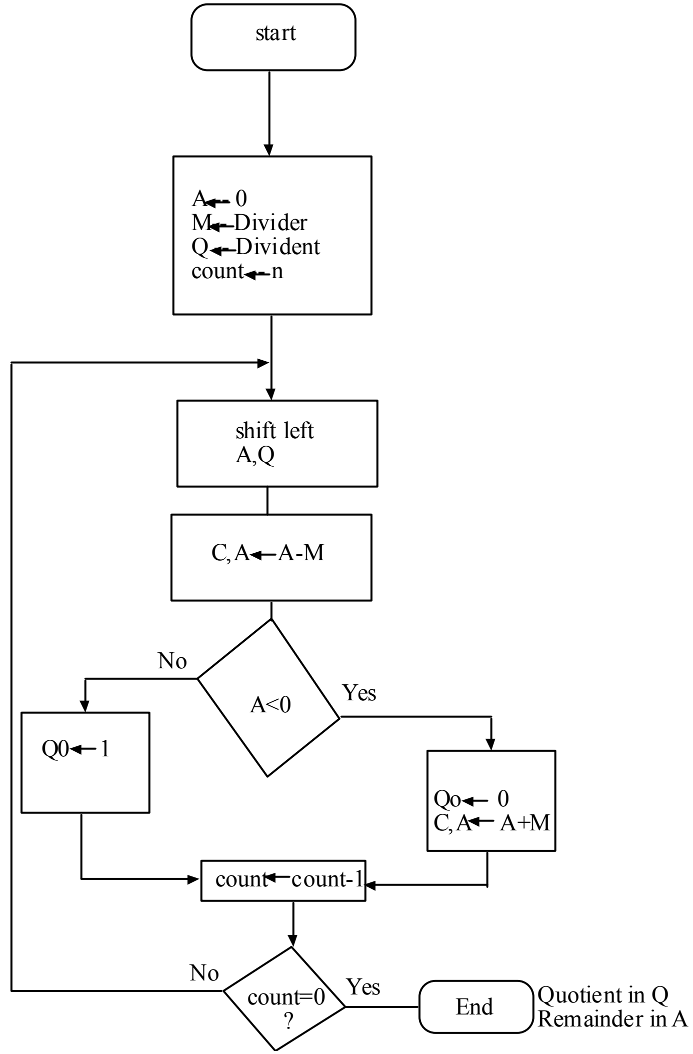


Fig : Flow chart for unsigned binary division.

A	Q	M = 0011
0000	0111	Initial value
0000	1110	Shift
1101		Subtractor
0000	1110	restore
0001	1100	shift
1110		subtractor
0001	1100	restore
0011	1000	shift
0000		subtractor
0000	1001	set Q_0 =1
0001	1001	shift
1110		subtractor
0001	0010	restore
(remainder) (1)	(quotient) (2)	

Fig: 7/3

The devisor is placed in M register, the dividend in the Q register at each step A and Q registers together are shifted to the left1 bit. M is subtracted from A to determine weather A divides the partial remainder. If it thus then Q_0 get 1 bit otherwise Q_o get 0 bit. And M must be added back to A to restore the previous value. The count is decremented and the process continuous for n steps. At the end the Quotient is in the Q register and remainder in the A register.

Date:2065/12/20

Floating point representation: The floating point representation of the number has two parts. The first part represents a signed fixed point numbers called mantissa. The second part designates the position of the decimal (or binary) point and is called exponent . For e.g the decimal no +6132.789 is represented in floating point with fraction and exponent as follows.

Fraction	exponent.
+0.6132789	+04

This representation is equivalent to the scientific notation $+0.6132789\times10^{+4}$

The floating point is always interpreted to represent a number in the following form $m\times r^e$.

Only the mantissa and the exponent e are physically represented in the register (including their sign) .The radix r and the radix point position of the mantissa are always assumed.

A floating point binary no is represented in similar manner except that it uses base 2 for the exponent.

For example the binary no +1001.11 is represented with 8 bit fraction and 0 bit exponent as follows.

0.1001110×2^{100}

Fraction	Exponent
01001110	000100

The fraction has zero in the leftmost position to denote positive. The floating point number is equivalent to $m \times2^e = +(0.1001110)_2 \times 2^{+4}$

Floating point arithmetic: The basic operation for floating point arithmetic are

<u>*Floating point number*</u>	<u>*Arithmetic Operations.*</u>
$X = x_s \times B^{xE}$	$x+Y = (x_s \times B^{XE-YE}+Y_s) \times B^{YE}$
$Y = Y_s \times B^{YE}$	$X-Y = (x_s \times B^{XE-YE}-Y_s) \times B^{YE}$
	$X*Y = (X_x\times Y_s) \times B^{XE+YE}$
	$X/Y = (X_s/Y_s) \times B^{XE-YE}$

For addition and subtraction it is necessary to ensure that both operands have same exponent value. This may require shifting the radix point on one of the operands to achieve alignment. Multiplication and division are more straight forward.

The exponent may be represented in biased exponent in this representation, the sign bit is remove from being separate entity. The bias is a positive no i.e added to the each exponent as floating point no is formed so that internally all exponents are positive. Consider an exponent that ranges form -50 to 49. It is represented in registers as positive nos. in the range of 0 to 99.

The register organization for floating point operation is shown in fig below.:

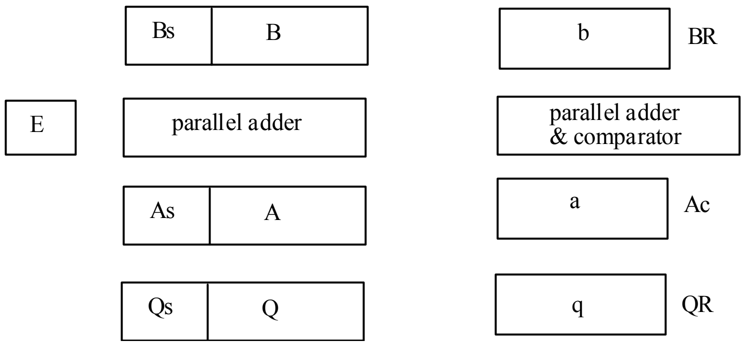


Fig: Register for floating point arithmetic operation.

There are two registers BR, AC and BR each register is subdivided into 2 parts . The mantissa has the uppercase letters symbols and the exponent part uses corresponding lowercase letters symbol.

It is assumed that each floating no has mantissa in sign magnitude representation and biased exponent. Note that the symbol AC represents the entire register that is concatenation of As A and a similarly register BR is subdivide into BS . B and b and QR into Qs, Q and q . A parallel adder adds the 2 mantissa and transfer the sum into A and carry into E, a separate parallel adder is used for exponent.

Addition and Subtraction: During addition and subtraction two floating point operands are in AC and BR. The sums or difference is formed in the AC. The algorithm can be divide into 4 consecutive parts.

1. Check for zeroes.
2. Allign the mantissa.
3. Add or subtract the mantissa.
4. Normalize the result.

* **Multiplication:** The multiplication can be subdived into 4 parts .

1. Check for zeroes .
2. Add the exponents.
3. Multiply mantissa.
4. Normalize the product.

Division: The division algorithm can be subdivided into 5 parts

1. Check for zeroes.
2. Initial registers and evaluate the sign.
3. Allign the dividend.
4. Subtract the exponent.
5. Divide the mantissa.

Date: 2065/12/25

Chapter: 4

Instruction set:

Machine instruction Characteristics:

The operation of the CPU is determine by the instruction it executes referred to as machine instruction or computer instruction. The collection of different instruction that the cpu can execute is referred to as CPU's instruction sets.

Each instruction must contain the information required by the CPU for execution. The elements of machine instruction are as follows:

1. Operation code.
   - Specifies the operation to be performed. (e.g ADD).
   - Source operand reference: Operands that are inputs for the operation.
   - Result operand reference: Operation may produce result.
   - Next instruction reference: This tells the CPU where to face the next instruction after the execution of this instruction is complete.

During instruction execution an instruction is read into the instruction register in the CPU. The CPU must be able to extract a data from various instruction field to perform the required operation.

It is difficult for both the programmer and the reader of text book to deal with binary representation of machine instruction. Thus it has become common practice to use symbolic representation of machine instruction.

Opcode are represented by abbreviations called mnemonics that indicates the operation. Common example include

ADD add

SUB Subtraction

MPY multiply

DIV divide

Operands are also represented symbolically. For example , instruction ADD R,Y add the value contain in data location y to the content of register R.

We can categories instruction types as follows:

1. Data processing: Arithmetic and logic instruction.
2. Data storage: memory instruction
3. Data movement: I/O instruction.
4. Control: Test and branch instruction.

Types of operands:

1. Address.
2. Number
3. Character.
4. Logical data.

Machine instruction operate on data. The most general categories of data are address, number, character and logical data.

Addresses are in fact a form of data in many cases some calculation must performed on the operand reference in a an instruction to determine the main memory address.

All machine languages include numeric data types. Three types of numerical data are common in computers.

- Integer or fixed point.
- Floating point.
- Decimal

Although all internal computer operation are binary in nature. The human user of the system deal with decimal number. Thus there is necessity to convert from decimal to binary on input and from binary to decimal on output.

A common form of data is text or character streams, a number of codes have been devised by which characters are represented by sequence of bits. The most commonly use character code in international reference alphabets (IRA) referred to in the Unites states as American standard code for information interchange (ASCII). Each charter in this code is represented by unique 7 bit patter. Thus 128 different character can be represented. Another code used to encode character is extended binary coded decimal interchange code (EBCDIC). It is 8 bit code in the case of EBCDIC 11110000 -11111001. Represent the digits zero through nine (0-9).

Normally each word or other addressable unit is treated as single unit of data it is sometimes useful, however to consider n bit unit as consisting of n one bit item of data. Each item having the value 0 and 1. when data are viewed this way they are considered to be logical data.

Date: 2065/12/26

Types of operations:

The number of different opcodes varies widely form machine to machine. However the same general types of operation are found on all machine. A useful and typical categorization is the following:

- Data transfer
- Arithmetic
- Logical
- Conversion
- I/O
- System control
- Transfer of control

Data transfer: The data transfer instruction must specify several things. First the location of source and destination operands must be specified. Each location could be memory register or the top of the stack. Second the length of data to be transfer must be indicated. Third as with all instruction with operands the mode of addressing for each operand must be specify. For example;

Operation name	**Description**
Move	Transfer word from source to destination.
Store	Transfer word from processor to memory.
Push	Transfer word from source to top of stack.
Pop	Transfer word form top of stack to destination

Arithmetic: Most machine provide the basic arithmetic operations of add , subtract , multiply and divide. Other possible operation include a variety of single operand instruction. For example;

Increment – Add one to the operand

Decrement – Subtract one from the operand.

Logical: Most machine also provide a variety of operation for manipulating individual bits of work. They are based upon Boolean operation. The basic logical operations that can perform on binary data are shown below:

P	Q	NOT P	P AND Q	P OR Q	P XOR Q	P=Q
0	0	1	0	0	0	1
0	1	1	0	1	1	0
1	0	0	0	1	1	0
1	1	0	1	1	0	1

Conversion: Conversion instruction are those that change format of data. An example is converting from decimal to binary.

Operation Name	**Description**
Convert	Convert the contents of word from one form to another.

Input/output : Input (read instruction) transfer the data form specified i/o port to the destination. O/P (write instruction) transfer data form specified source to i/o port.

System control: These instructions are reserved for the use of operating system. A system control instruction may read or altered control register.

Transfer of control: For all of the operation types discussed so far, the next instruction to be performed is the one that immediately follows in memory the current instruction. How ever a significant fraction instruction in any program have as their function changing the sequence of instruction execution.

Assembly language: A CPU can understand and execute machine instruction. Such instruction are simply binary numbers stored in the computer. If a programmer wished to program directly in machine language , then it would be necessary to inter the program as binary data.

Consider the statement N = I+J+K. Suppose we wished to program this statement in machine language and to initialize the I,j and k to 2,3 and 4 respectively. The program starts in location 101(hexadecimal). Memory is reserved for four variable starting at location 201. The program consists of 4 instructions.

1. load the content of location 201 into the Ac.
2. At the content of location 202 to the Ac.
3. At the content of location 203 to the Ac.
4. Store the content of Ac in the location 204.

Address	**Contents:**	
101	0010 0010 0000 0001	(2201)
102	0001 0010 0000 0010	(1202)
103	0001 0010 0000 0011	(1203)
104	0011 0010 0000 0100	(3204)
201	0000 0000 0000 0010	(0002)
202	0000 0000 0000 0011	(0003)
203	0000 0000 0000 0100	(0004)
204	0000 0000 0000 0000	(0000)

Address	Instructions
101	LDA 201
102	ADD 202
103	ADD 203
104	STA 204
201	DAT 2
202	DAT 3
203	DAT 4
204	DAT 0.

(c) Symbolic program

Label	Operation	Operand
FORMUL	LDA	I
	ADD	J
	ADD	K
	STA	N
I	DATA	2
J	DATA	3
K	DATA	4
N	DATA	0

(d) Assembly language.

A slight improvement is to write the program in hexadecimal rather than binary notation. For improvement we can make use of symbolic name or mnemonic of each instruction. With the last refinement we have assembly language. Program written in assembly language are translated into machine language by a assembler. This program must not only do the symbolic translation but also assign some form of memory address to symbolic address.

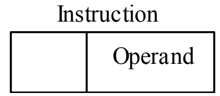
Date: 2065/12/27

Addressing:

The most common addressing techniques are:

- Immediate
- Direct
- Indirect
- Register
- Register indirect
- Displacement
- Stack

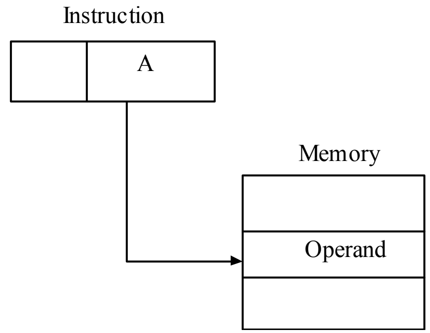
Immediate addressing: The simplest form of addressing is immediate addressing in which the operand is actually preset in the instruction.



This mode can be used to define and use constant or set initial value of the variable.

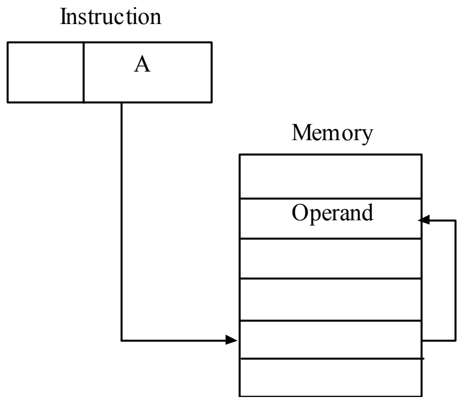
Direct addressing: A very simple form of addressing is direct addressing in which the address filed contains the effective address of the operand. EA = A

EA – Effective address of the location containing reference operand.

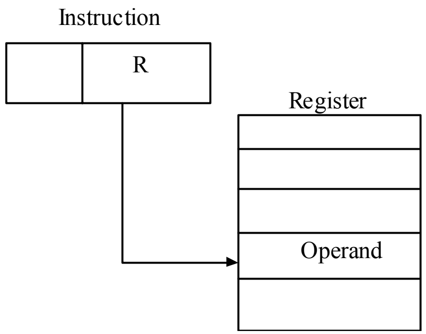


Indirect addressing: With the indirect addressing the length of addressing field is less than the word length thus limiting the address length. One solution is to have the address field referred to address of a word in memory which in term contains full length address of the operand. This is know as invalid addressing.

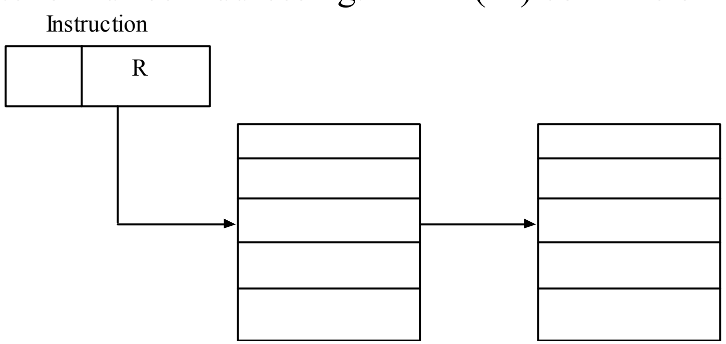
EA = (A)← contains of A



Register addressing: It is similar to direct addressing. The only difference is that, the address field refers to register rather than the main memory address.

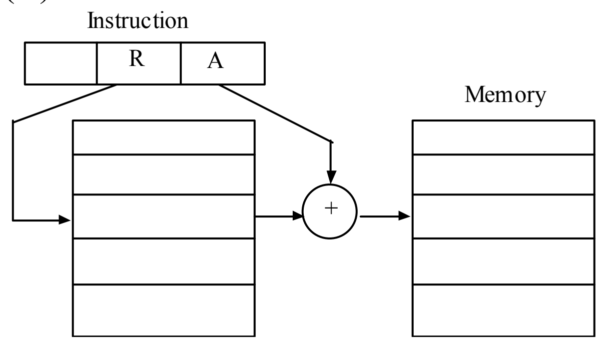


Register indirect addressing: Register indirect addressing is analogous to indirect addressing. EA = (R) contains of R.

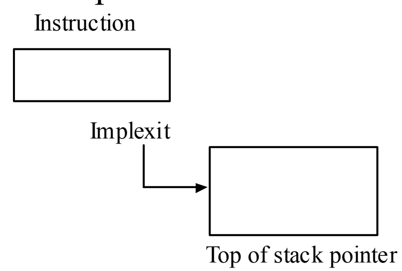


Displacement addressing: A very powerful mode of addressing combines the capabilities of direct addressing and register indirect addressing.

FA = A+(R)



Stack addressing: The stack is the linear array of locations. It is some times referred to as push down list or last in First out (LIFO) queue. The stack pointer is maintained in register.



Instruction Format: An instruction format must include opcode and implicitly or explicitly zero or more operands.

The most basic design issue to be faced is the instruction format length. This decision affects and is affected by memory size, memory organization bus structure, CPU complexicity and CU Speed. More opcodes and more operands makes like easier for a programmar because shorter program can be written to

accomplish a given task. All of these things (opcodes, operands, address range) require bits and push in the direction of longer instruction length. But longer instruction length may be wasteful . A 64 bit instruction occupies twice the space of 32 bit instruction. But is probably less than twice as useful.

An equally difficult issue is how to allocate the bits in that format. For a given instruction length there is clearly trade off no of opcodes and the power of addressing capabilities. More opcodes obviously mean more bits in the opcode field, for an instruction format of given length. This reduces the no of fields available for addressing. This is the interesting refinement to this trade off and that is use of variable length opcodes.

Date:2066/1/3

5. **CPU structure and Function:**

Processor organization: To understand the organization of CPU. Let us consider the requirements placed on the CPU. The things that is must do :

- *fetch instruction:* CPU reads instruction form memery.
- *Interpret:* The instruction is decoded to determine what action is required.
- *Fetch data:* The execution of an instruction may require reading data form memory or I/O module.
- *Process data:* The execution of an instruction may require performing some arithmetic or logical operation on data.
- *Write Data:* the result of an execution may require writing data to the memory of I/O module.

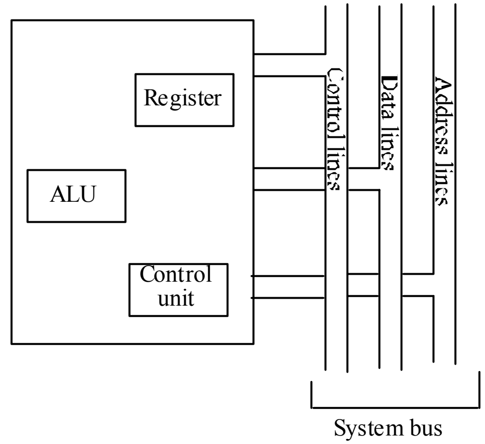


Fig: CPU with system bus.

Fig shows simplified view of CPU indicating its connection to the rest of the system via system bus. The major components of CPU are ALU and control unit in addition the fig shows a minimum internal memory consisting set of storage location called register.

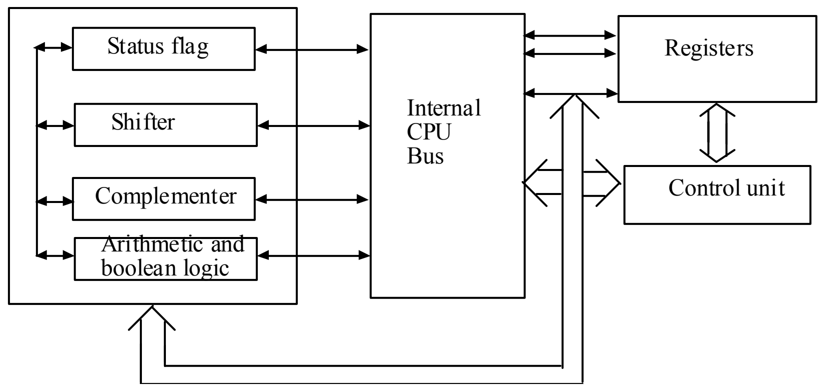


Fig: internal structure of CPU.

Figure shows more detail view of CPU. The data transfer and logic control path are indicated including internal CPU bus.

Register organization: Within the CPU there is the set of registers that functions as level of memory above main memory and cache in the hierarchy. The register and CPU perform two rolls:

1. User visible register: These enables the machine or assembly language programmer to minimize main memory references by optimizing use of register.
2. Control and status register: These are use by the control unit to control the operation of CPU.

Types of user visible register: A user visible register is one that may be referenced by mean of machine language that the CPU executes. We can characterized these in the following categories.

1. General purpose
2. Data register
3. Address register
4. condition code register.

General purpose register can be assigned to a variety of function by the programmer. Some times they are use within the instruction set is orthogonal to the operation i.e any general purpose register can contain the operand for any opcode.

Data register may be used only to hold data and can not be employed in the calculation of operand address.

Address register may themselves be some what general purpose or they may be devoted to a particular addressing mode.

A final categorize of register which at least partially visible to the user holds condition code (flags). Condition codes are bits set by the CPU as the result of operation . For example , arithmetic operation may produce +ve , -Ve , zero or overflow result. In addition to the result itself being stored in the register or memory a condition code is also set. Condition code bit are collected into one or more registers usually they form part of uncode register.

Control and status register: There are variety of CPU register that are employed to control the operation of CPU. Four register are essential to instruction execution:

- Program counter (PC)
- Instruction register (IR)
- Memory address register (MAR)
- Memory buffer register(MBR).

These four register are use for the movement of data between the CPU and memory.

All CPU designs include a register or set of register often known as program status word(PSW). PSW typically contain condition code pulse other status information. Common flags includes the following:

1. Sign: Sing contain the sign bit of result of arithmetic operation.
2. zero : Set when the result is zero.
3. Carrey: Set if operation resulted in Carrey into or borrow out of the higher order bit.
4. Equal: Set if a logical compare result is equality.
5. Overflow: Used to indicate arithmetic overflow.
6. Interrupt enable disable : used to enabled or disable interrupt.

Instruction cycle:

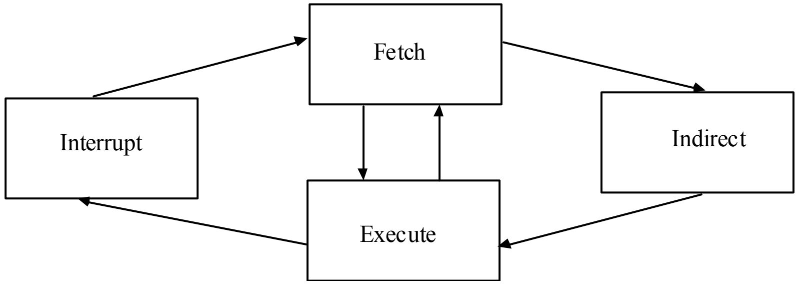


Fig: Instruction cycle:

The execution of an instruction may involve one or more operands in memory each of which requires a memory access. Further it indirect addressing is used then additional memory access are required.

We can think of fetching of indirect address as one more instruction subcycle. The main line of activity consists of alternating instruction fetch and instruction execution activities. After an instruction is fetched it is examine to determine if any indirect addressing is involved. If so required operations are fetched using indirect addressing. Following execution and interrupt may be processed before the next instruction fetched.

During fetch cycle an instruction is read form the memory. Figure shows flow of data during this cycle.

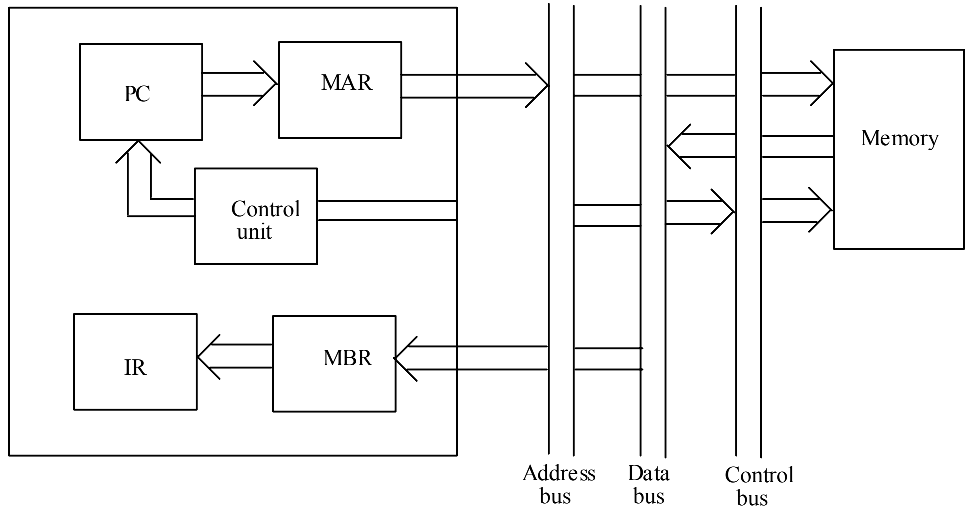


Figure: Data flow, fetch cycle.

The program counter contains the address of next instruction to be fetched. This address is moved to the MAR and placed on the address bus. The control unit request the memory read and the result is placed on the data bus and copied into the MBR and then move to the IR. Mean while the PC is incremented by 1.

Once the fetched cycle is over. The control unit examine the contains of IR to determine if it contains operand specifier using indirect addressing. If so indirect cycle is performed.

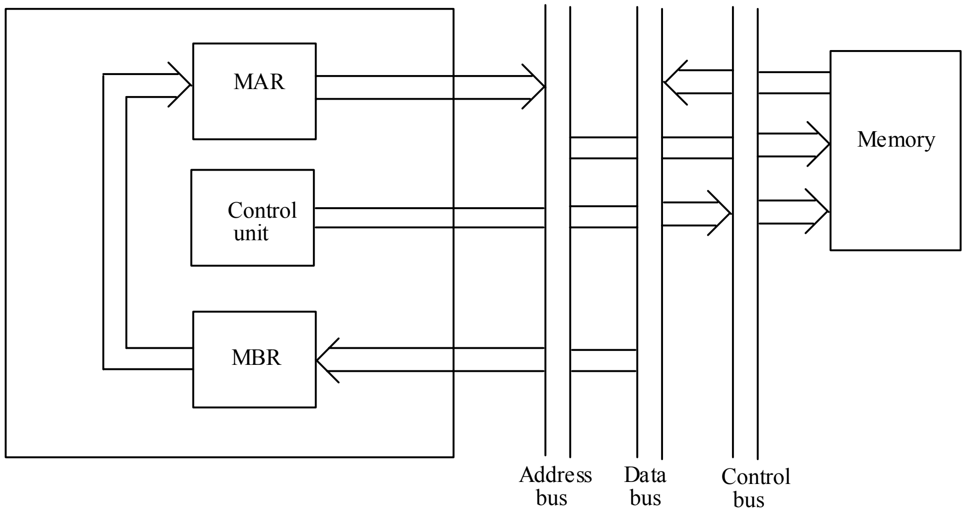


Fig: Data flow, Indirect cycle

The right most N bits of MBR which contains the address reference are transfer to the MAR then the control unit request the memory read to get the desire address of operand into the MBR.

The fetch and indirect cycle are simple and predictable. The execute cycle takes many forms, the forms depends on which of the various machine instruction is in IR. This cycle may involve transferring data among registers, read or write from memory or i/o.

Like fetch and indirect cycle, interrupt cycle is simple and predictable.

Date:2066/1/9

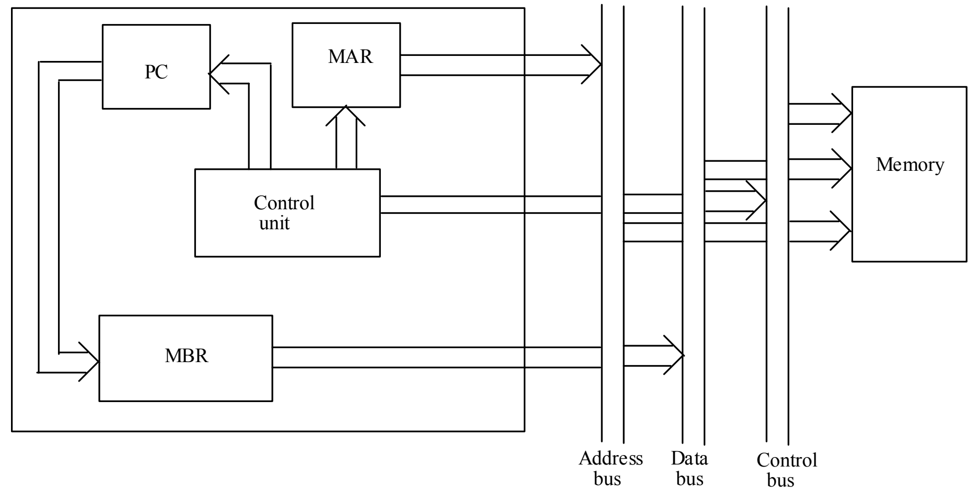


Fig: Data flow interrupt cycle.

The current contents of PC must be set, so that the CPU can be resume normal activity after the interrupt. Thus the content of PC are transfer to the MBR to be written in the memory. The special memory location reserve for this purpose is loaded into the MAR from control unit. The PC is loaded with the address of interrupt routine.

Instruction pipelining: As a simple approach, consider subdividing instruction processing into two stages: fetch instruction and execution instruction. There are times times during the execution of instruction when main memory not being access this time could be use to fetch next instruction in parallel with the execution of current one. Fig explain this approach.

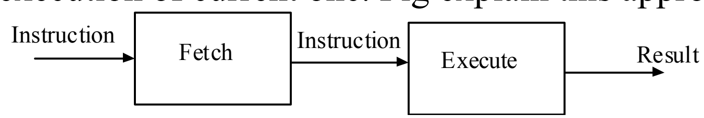


Fig: Two stage Instruction pipelining.

The pipe line has two independent stages. The first stage fetches an instruction and buffers it when the second stage is free the first stage passes it the buffer instruction. While the second stage is executing the instruction the first stage takes advantage of any unused memory cycles to fetch and buffer the next instruction. This is called instruction prefetch or fetch overlape. This process will speed up instruction execution.

To gain further speed the pipe line must have more stages. Let us consider the following decomposition of instruction processing:

- Fetch instruction (FI)
- Decode instruction (DI): Determine the upcode and operand specifies.
- Calculate operand(CO): calculate the effective address of each source operand.
- Fetch operand (FO): Fetch is operand from memory.
- Execute instruction(EI): Perform the indicated operation.
- Write operand(WO): Store the result in memory.

With this decomposition the various stages will be of more nearly equal duration for the sake of illustration let us assume equal duration. Using this assumption figure shows that six stage pipe line can be reduced the execution time for five instruction from 30 time units to 10 time units.

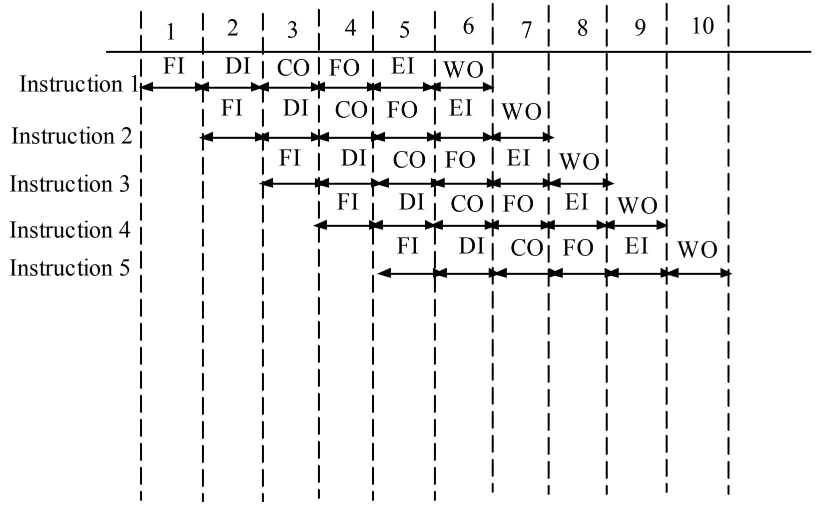


Fig: Timing diagram for instruction pipelining operation.

Several comments are in order:

Diagram assumes that each instruction goes through all six stages of pipeline. This will not always be the case. For example, load instruction doesn't need WO stage however to simplify the pipeline hardware, the timing is set up assuming that each instruction requires all six stages. Also the diagram assumes that all of the stages can be performed in parallel. In particular it is assume that there is no memory conflict. For example, FI, FO and WO stages involve memory access. The diagram implies that all these access can occur simultaneously. Most memory system will not permit that. How ever the desired value may in cache or FO or WO stage may be null. Thus much of the time memory conflict will not slow down the pipeline.

Several other factor serve to limit the performance enhancement. If the six stages are not of equal duration there will be some waiting involve at various pipeline stages. Other difficulties, the condition branch instruction can invalidate

several instruction fetches. A similar unpredictable event is interrupt.

Assume that instruction 3 is the conditional branch to instruction 15. Until the instruction is executed there is no way of knowing which instruction will come next. The pipe line in this example simply load the next instruction in sequence (instruction 4)and proceeds.

Date: 2066/1/10

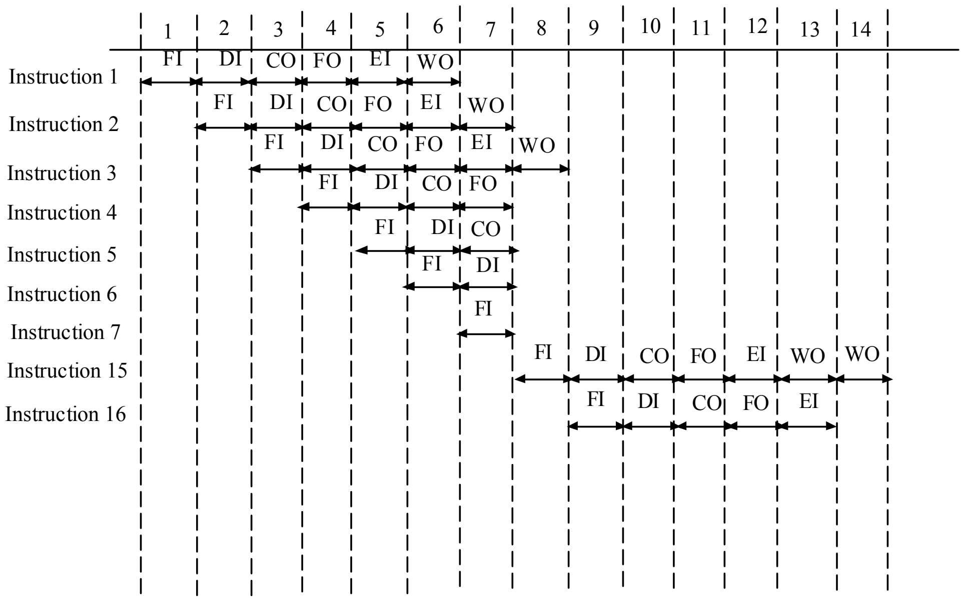


Fig: Effect of conditional branch on instruction pipeline instruction.

In the figure the branch is taken. This is not determine until the end of time unit 7. At this point the pipe line must be cleared of instruction that are not useful. During item unit 8 the instruction 15 enters the pipeline. No instruction complete during the time units 9-12. This is the performance penalty incurred because we couldn't anticipate the branch.

Figure indicates the logic needed for pipelining to accounts for branches and interrupts.

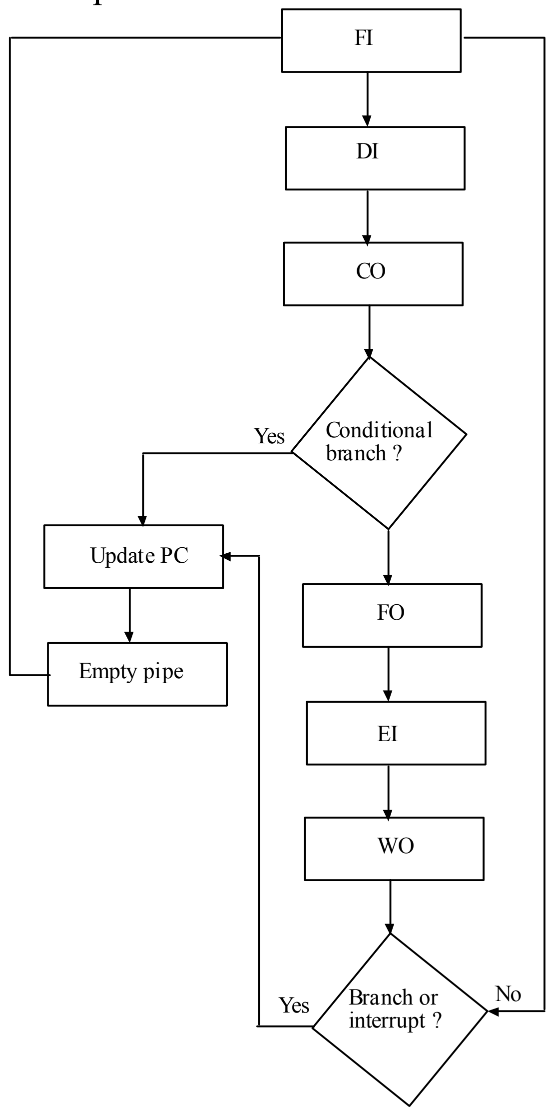


Fig: six stage CPU instruction pipeline.

The Pentium Processor:

Register organization:

The register organization include the following type of register:

General: There are eight 32 bit register. This may be used for all type of Pentium instruction. They can also hold operand for address calculation.

Segment: There are six 16 bit segment register. The code segment register references the segment containing the instruction being executed. The stack segment register references the segment containing the user visible stack.

Flags: It includes six condition codes (carry, parity, auxiliary, zero, sign, overflow). Which report the results of integer operation. In addition there are bits in the register that may be referred to as control bits. Interrupt enable flags when set the processor will recognize external interrupt.

Instruction Pointer: It contains the address of instruction.

Control register: The Pentium employs four 32 bit control registers. Control various aspect of processor operation.

There are also register specifically devoted to the floating point unit:

Numeric: Each registers holds extended precision 80 bit floating point number.

Control: 16 bit control register contains bits that control the operation of floating point unit.

Status: 16 bit status register contains the bits that reflect the current state of floating point unit.

Tag word: 16 bit register contains 2 bit tag for each floating point numeric register, Which indicates the nature of contents of corresponding register. The four possible values are valid, zero, special (infinity) and empty.

The power PC organization:

Register organization: The fixed point unit includes

General: There are 32 sixty four bit general purpose registers. These may be used to load, store and manipulate data operands and may also used for register indirect addressing.

Accept ional register: Includes 3 bit that repot exceptions in integer arithmetic operations.

The floating point unit includes addition user visible register

General: There are 32 sixty four bit general purpose register used for all floating point operation.

Floating pint status and control register: This 32 bit register contains bits that control the operation of floating point unit and bits that record status resulting form floating point operation.

The branch processing unit contains user visible registers.

Condition register: Consists of 8 four bit condition code.

Link register: This register is used for call/return instructions . If the LK bit in condition branch instruction is set then the address following the branch instruction is fetched in the link register and it can be used for later return.

Count: The count register can be used to control the iteration loop. The count register is decremented each time, it is tested in conditional branch instruction.

Chapter: 6

Instruction execution characteristics: One of the most visible form of evaluation associated with a computer is that of programming languages. The response from researches and industry has been to develop ever more powerful and complex high level programming languages. These high level languages allow the programmer to express algorithm more concisely take care of much of the detail and often support naturally the use of structure programming or object oriented design.

This solution give rise to another program known as symmetric gap, difference between the operation provided in HLL and those provided in computer architecture. Symptom of these gap are execution inefficiency excessive machine program size and compiler complexicity. Designers responded with architectures intended to close this gap. Key feature includes large instruction sets dozens of addressing modes and various HLL statements implemented in hardware. Such complex instruction set are intended to:

- Ease the task of complier writer.
- Improve Execution efficiency.
- Provides support for even more complex and suffocated HLL.
- Mean while and number of studies have been done over the years to determine the characteristics and patterns of execution of machine instruction generated from HLL program. The results of these studies inspire some researchers to look for a different approach namely to make the architecture that support the HLL simpler rather than more complex.

Use of large register file:

The result summarize in instruction execution characteristics point out the desirability of quick access to operand. We have seen that there is large proportion of assignment statement in HLL program and many of this are simple form A←B also there is significant no of operand access per HLL statement. If we coupled these result with the fact that most accesses are to local scalarors. Heavy reliance or register storage is suggested.

The reason that the register storage is indicated is that it is the faster available storage device faster than both main memory and cache. The register file is physically small on the same chip as ALU and control unit. Thus the strategy is needed that will allow most frequently access operand to be keep in register and minimize register memory operation.

Two basic approach are possible, one based on hardware and other on software.

The software approach is to reli complier to maximize register usages. The complier is attempt to allocate register to those variable that will be used to most in a given time period. The hardware approaches is simple to used more register so that more variable can be held in the register for longer period of time.

Register window:

On the face of it, the use of large set of register should decrease the need to access memory, because the most of operand references are to local scalars the obvious approach is to store these in the register with perhaps with few register reserved for global variable. The problem is that the definiatatio of local changes with each procedure call and return operation that occur frequently . On every call local viable must be saved from the

register input memory so that the register can be reused by call programs. Further more the parameter must be passed on return the variables of the parent program must be restore and results passed back to the parent program.

The window register is divided into fixed size areas, parameter register hold the parameters passed down form the procedure that called the current procedure and hold the results to be passed back up. Local registers are used for local variables as assigned by compiler, temporary registers are used to exchange parameters and results with the next lower level. (procedure called by current procedure).

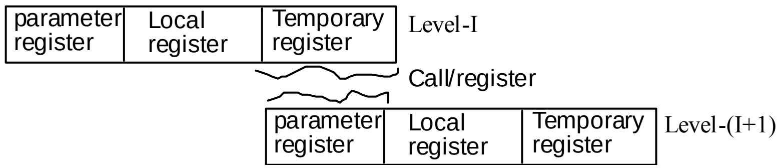


Fig: overlapping register.

Complier-Based register optimization:-

The objective of complier is to keep the operands for as many computation as possible in registers rather then main memory and to minimize the load and store operations .

Each program quantity that is a candidate for residing a registers in assigned to a symbolic register or virtual registers. The complier then maps the unlimited no of symbolic into a fixed no of real registers. Symbolic registers whose uses doesn't overlap can share the same real registers. If in a particular portion of a program there are more quantities to deal with than real registers then reuse of the quantities are assigned to memory locations. Load and store register are used to position quantities in registers temporarily for computational operations.

The essence of the optimization task is to decide which quantities are to be assigned to registers at any given point in th program. The technique must commonly used is known as graph coloring.

Given a graph consisting of nodes and edges assign color to node such that adjacent modes have different colors and do this in such a way as to minimize the no of different color. First the program is analyzed to build a register interference graph the nodes of the graph are symbolic registers if two symbolic registers are live during the same program framgment then they are joined by edge to depict the interference. An attempt is then made to color the graph with n colors where 'n' is the no of registers nodes that share the same color can be assigned to same registers if this process doesn't fully succeed than those nodes that can't be colored must be placed in memory. Assume a program with six symbolic registers to be complied into three actual registers.

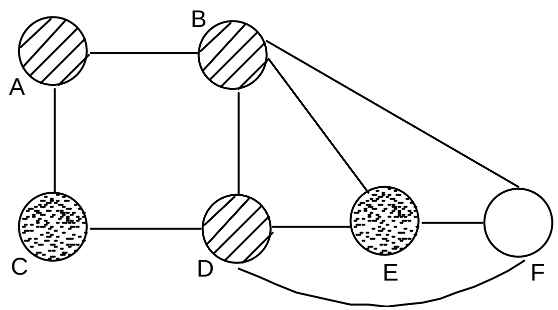


Fig: Register interface graph.

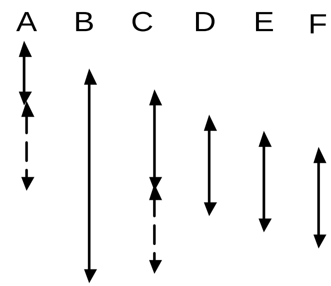


Fig: timing sequence of active use of register.

Reduced instruction set computer (RISC) characteristics:

In the early 1980's a number of computer designers recommended that computer use fewer instructions with a simple construct so they can be executed much faster within the CPU without having to use memory as often. This type of computer is classified as RISC. The concept of RISC architecture involves attempt to reduce execution time by simplifying the instruction set of the computer. The major characteristics of RISC processor are:

1. Relatively few instruction.
2. Relatively few addressing mode.
3. Memory access limited to load and store instruction.
4. All operations done within the register of CPU.
5. Fixed length, easily decoded instruction format.
6. Single cycle instruction execution.
7. Hard-wired rather than micro program control

Complex Instruction set computer (CISC) characteristics:-

A computer with large no of instruction is classified as complex instruction set computer (CISC). The major characteristics of CISC architecture are :

1. Large no of instructions typically form hundred to 250 instructions.
2. Same instructions that perform specialized task and are used in frequency.
3. A large variety of addressing modes typically form 2-50 different modes.
4. Variable length instruction format.
5. Instruction that manipulate operands in main memory.

RISC Pipelining:

The simplicity of the instruction set can be utilized to implement the instruction pipeline using a small no of sub-operations with each being executed in one clock cycle. All data manipulation instructions have register to register operations. Since all operands are in register there is no need of calculating the effective address or fetching of operand from memory. The instruction cycle can be divided into 3 sub-operations and implemented in 3 segments:

1. I - Instruction fetch.
2. A – ALU operation.
3. E - Execute instruction.

Consider now the operation of following four instruction: -

1. Load $R_1 \leftarrow M$ [address 1]
2. LOSD:$R_2 \leftarrow M$ [address 2]
3. ADD:$R_3 \leftarrow R_1+R_2$
4. STORE:M[address] $\leftarrow R_3$

If 3 segment pipeline proceeds without interrupt ther willl be data conflict in instruction three because the operand in R_2 is not yet available in A segment. This can be seen from the timing of pipeline shown in fig.

Clock cycles	1	2	3	4	5	6
1. Load R_1	I	A	E			
2. Load R_2		I	A	E		
3. Load R_1+R_2			I	A	E	
4. Load R_3				I	A	E

a) Pipeline with data conflict.

The E segment in clock cycle '4' is in the process of placing the memory data into R_2. The A segment in clock cycle 4 is using the data from R_2 but the value in R_2 will not be the correct value

since it has not yet been transferred from memory. If compiler can not find a useful instruction to put after the load it inserts no operation instruction thus wasting a clock cycle. This concept of delaying the use of data loaded form memory is referred to as delayed load.

Clock cycles	1	2	3	4	5	6	7
1. Load R_1	I	A	E				
2. Load R_2		I	A	E			
3. No operation			I	A	E		
4. Add R_1 +R_2				I	A	E	
5. Store R_3					I	A	E

Date: 2066/1/22
Chapter:7
Control unit and micorprogrammed control

Micro-operation.
The operation of computer is executing a program consists of sequence of instruction cycle. Each instruction cycle is made up off no of smaller units, one subdivision that we found convienent is fetch, indirect execute and interrupt with only fetch and execute cycle always occurring. Each of the smaller cycle involve series of steps, each of which involve processor register. We will refer to these steps as micro operations. Fig depict the relationship among the various concepts we have been discussing.

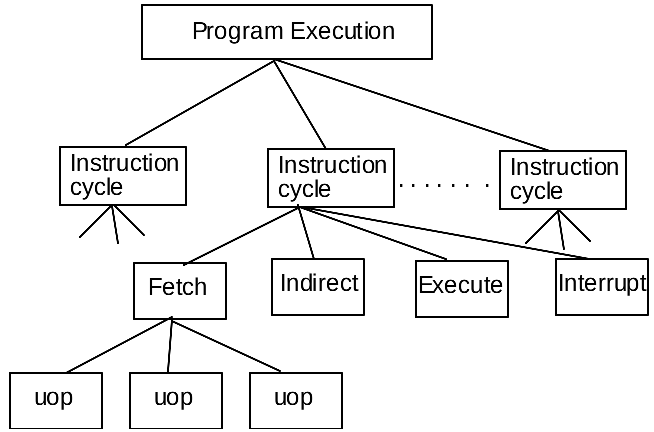


Fig: constituent element of program execution.

Fetch cycle: It causes an instruction to be fetched form memory. Fetch cycle actually consist of three steps and four micro operations.

t1: MAR $\leftarrow$ (PC)
t2: MBR $\leftarrow$Memory
PC $\leftarrow$PC+1
t3: IR $\leftarrow$(MBR)

The notion (t1,t2,t3) represent successive time units.

Indirect cycle: once an instruction is fetch , the next step is to fetch source operand.

t1: MAR $\leftarrow$ (IR (address))
t2: MBR $\leftarrow$Memory
t3: IR(address) $\leftarrow$(MBR(address))

Interrupt cycle: At the completion of execution cycle a test is made to determine weather any enabled interrupts have occur if so the interrupt cycle occurs.

t1: MBR $\leftarrow$ (PC)

t2: MAR ←save address
PC ←Routine address
t3: Memory ←(MBR)

Execute cycle: The fetch indirect and interrupt cycle are simple and predictable. Each involve fix sequence of micro operation. This is not true of the execute cycle for a machine with N different upcodes, there are N different sequence of micro operation that can occur. Consider ADD instruction.

ADD R1,X

Which adds the content of location X to register R1.

t1: MAR ← (IR address)
t2: MBR ←Memory
t3: R1 ←(R1)+(MBR)

Date: 2066/1/24

Control of processor:

We can define the functional requirements for the control unit. A definition of these functional requirement is the basis for design and implementation of the control unit. The following three steps process lead to characterization of control unit.

1. Define the basic elements of the processor
2. Describe the micro operation that the processor performs.
3. Determine the functions that the control unit must perform to cause the micro operations to be performed.

The basic functional elements of processor are:

- ALU
- Register.
- Internal data path.
- External data path.
- Control unit
-

All micro operation fall into of the following category.

- Transfer data from one register to another.
- Transfer data from one register to external interface.
- Transfer data from external interface to register.
- Perform the arithmetic or logic operation using register for input and output.

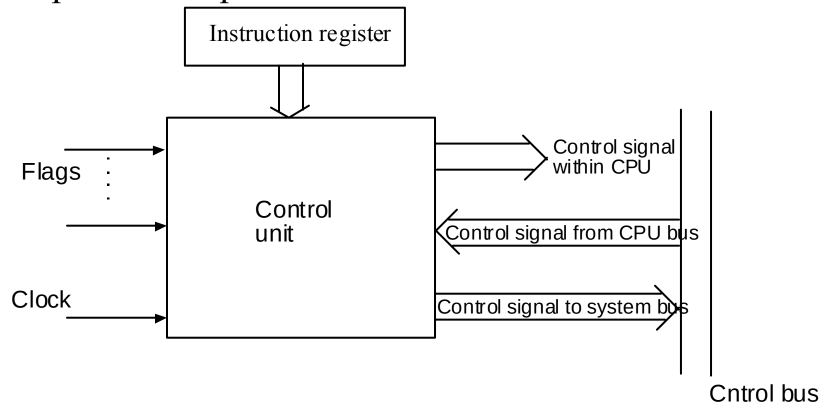


Fig: model of control unit.

Hardwire implementation: In hardwire implementation the control unit is essentially a combinatorial circuit. Its input logic signal are transform into set of output logic signal which are the control signal.

The key inputs are instruction register, clock , flag and control bus signal. The control unit makes the use of op-code and will perform the different actions for different instructions. To simplify the control unit logic, there should be unique logic input for each op-code. This function can be performed by decoder which takes encoded input and produces and single output.

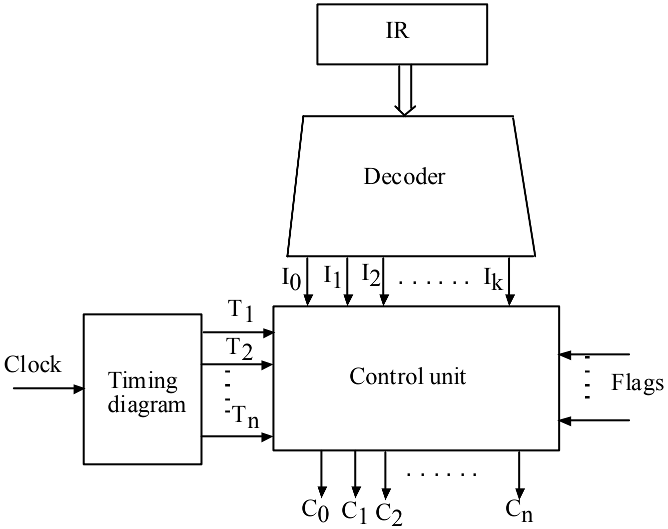


Fig: Control unit with decoded input.

Let us consider a single control signal c5. This signal causes data to be read from external data bus into MBR. Let us define two new control signal P and Q that have following interpretation.
PQ = 00 Fetch cycle.
PQ = 01 Indirect cycle .
PQ = 10 Execute cycle .
PQ = 11 Interrupt cycle.

Then the following Boolean expression define c5.
$C5 = P'Q'T_2 + p'.Q.T_2$
i.e the control signal c5 will be asserted during the 2^{nd} time unit of both fetch and indirect cycle.

Microinstruction sequencing: The two basic task performed by micro programmed control unit are as follows:

- Micro instruction sequencing:- Get then next micro instruction from the control memory.
- Micro instruction execution:- Generates the control signals needed to execute the micro instruction.

Based on the current micro operation , condition flags and content of instruction register, control memory address must be generated for next micro instruction. A wide variety of techniques have been used. We can group them into three general categories based on the format of address information in the micro instruction:

- Two address field.
- Single address field.
- Variable format.

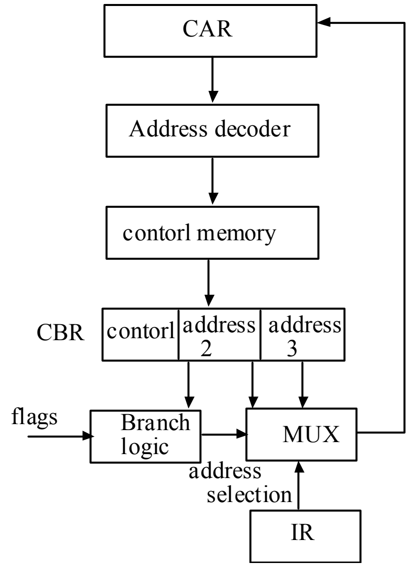


Fig: branch control logic, two address fields.

A multiplexer is provided that serves as destination for both address field plus instruction register based on the address

selection input the multiplexer transmits the op-code or one of the two address to the control address register (CAR). CAR is subsequently decoded to produce the next micro instruction address.

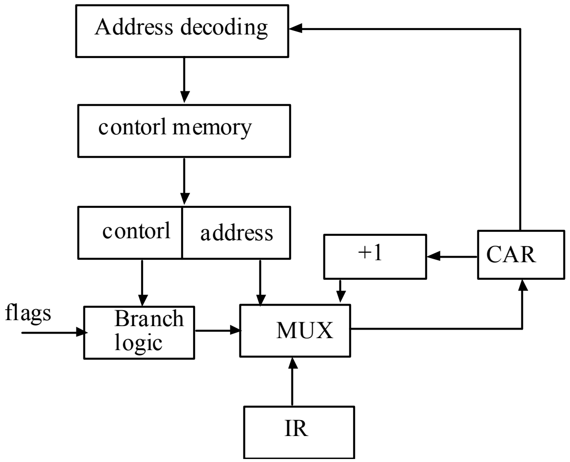


Fig: Branch control logic single address field.

Micro instruction execution:

The effect of execution of micro instruction is to generate control signal. Some of these signals control points internal to the processor. The remaining signal go to the external control bus.

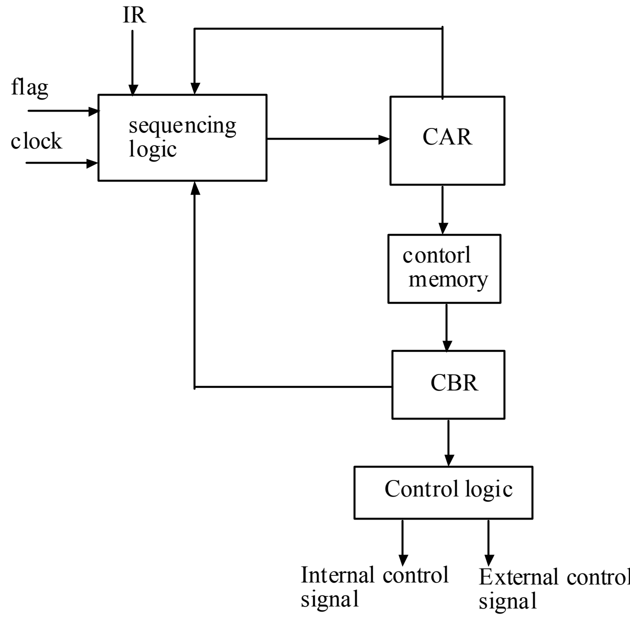


Fig: control unit organization.

The sequencing logic module generates a address of next micro instruction using as inputs instruction register flags, CAR(for implementing), control buffer register. The module is driven by clock that determine the timing of micro instruction cycle. The control logic module generate the control signal as a function of some of the bits in micro instruction.

Application of microprogramming:

The set of current application for micro programming include:

- Realization of computer.
- Micro program approach offer a systematic technique for control unit implementation. A relative technique is emulation. Emulation refer to used of microprogramming on one machine to execute program original written for another.

- Another used of microprogram is in the area of operating system supports.
- Realization of special purpose device a good example of this is data communication bore
- High level language support microprogramming can be used to support monitoring detection, isolation and repair of system error. These feature are known as micro diagnostics and significantly enhance the system maintenance facility.
- User tailoring, a no of machine produced writable control store that is control memory implemented in RAM rather than ROM and allows the user to write micro programs. These allows the user to tailor the machine to the desire application.

Date:2066/1/29

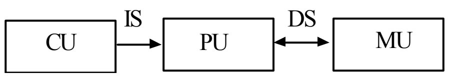
Chapter:- 8

Parallel organization:-

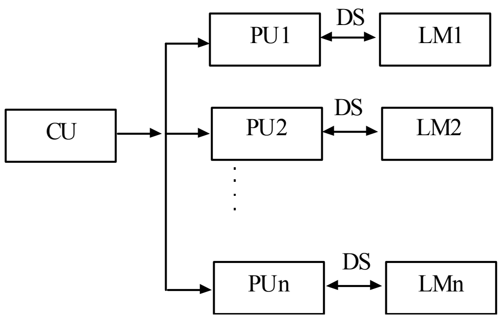
Parallel processor system:-

The most common way of categorizing computer system are:

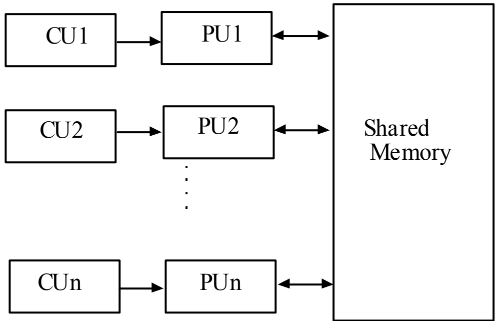
1. *Single instruction single data stream(SISD)*: A single processor executes a single instruction stream to operate on data stored in single memory.
2. *Single instruction multiple data (SIMD) stream*:- A single machine instruction controls the simultaneous execution of no of processing elements. Each processing element elements has associated data memory so that each instruction is executed on different set of data by different processes.
3. *Multiple instruction single data(MISD)stream:-* A sequence of data is transmitted to a set of processor. Each of which executes different instruction sequence. This structure is not commercially implemented.
4. *Multiple instruction multiple data(MIMD) stream:-* A set processor simultaneously execute different instruction sequence on different data set.



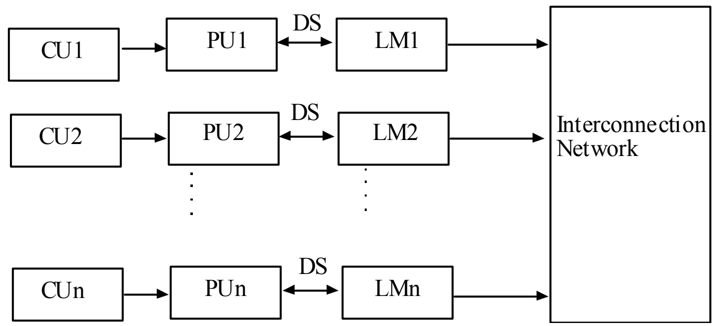
a) SISD



b) SIMD(with distributed memory)



c) MIMD (with shared memory)



d) MIMD(with distributed memory)

Fig: Alternative computer organization.
CU = control unit.
PU= processing unit.
IS = Instruction stream.
DS= Data stream.
MU= Memory unit.
LM= Loosely packed memory (distributed memory)
TM= trgutly packed memory.

With SISD there is some sort of control unit (shared memory) that provide instruction string to processing unit. The processing unit operate on single data stream from memory unit. With SIMD there is single control unit, now feding single instruction unit to multiple processing unit. Each PU may have its own dedicated memory or there may be a shred memory. Finally with MIMD there are multiple control units each feeding a separate instruction stream to its own PU. The MIMD may be sheared memory multiple processor or distributed memory multiprocessor.

Multiprocessing: A multiprocessor system is interconnection system of two more CPU with memory and I/O equipment. Multiprocessor are classified as multiple instruction multiple data string (MIMS). Multiprocessing improves the reliability of the system so that filer or error in one part has limited effect on rest of the system. If a fault causes one processor to fail, second processor can be assign to perform the disabled processor.

The benefit derived from multiprocessor organization is include system performance. The system derives its high performance from the fact that computation can proceed in parallel in one of the two ways.

1. Multiple independent jobs can be made to operate in parallel.
2. A single job can partition in to multiple parallel task.

The interconnection between the components of multiprocessor can have different physical configuration depending on the number of transfer path that are available between the processor and memory. Some of them are:

1. Time shared common bus.
2. Multi port memory.
3. Crossbar switch.

Time shared common bus:

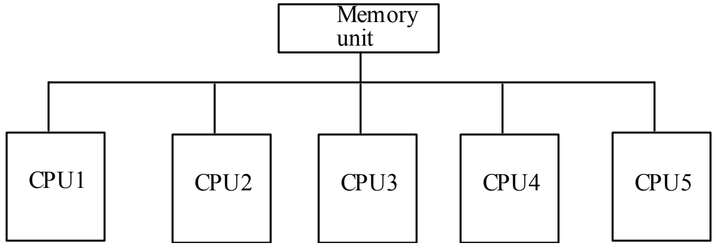


Fig: Time shared common bus organization.

A common bus multiprocessor system consist of number of processor connected through common path to a memory unit. A time shear common bus for 5 five processor is shown in fig . Only one processor can communicate with memory or another processor at a given time.

Multiport memory:

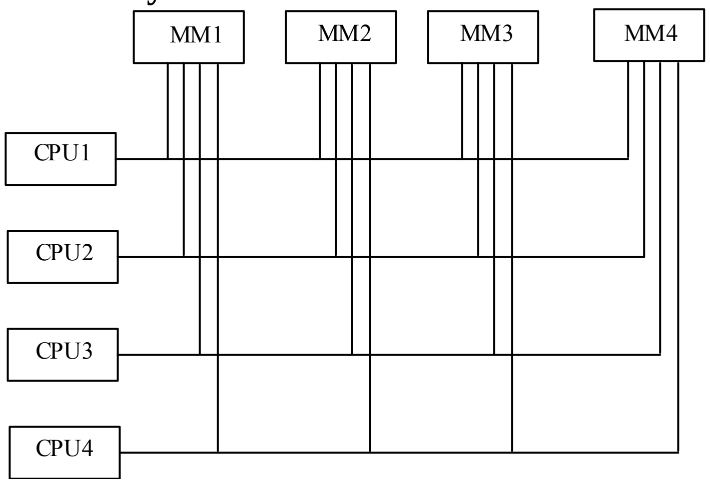


Fig: multiport memory organization.

A multiple memory system employs separate buses between each memory module and each CPU. This is shown in figure for four CPU and four memory module. Each processor bus is connected to each memory module. The memory module is said to have four ports and each ports accommodates one of the buses. The module must have internal control logic to determine which port will have to access to memory at any given time. Memory access conflict results are reserve by assigning fixed priority to each memory ports. Thus CPU1 will have priority over CPU2, CPU2 will have priority over CPU3, and CPU4 will have lowest priority.

Crossbar switch:-

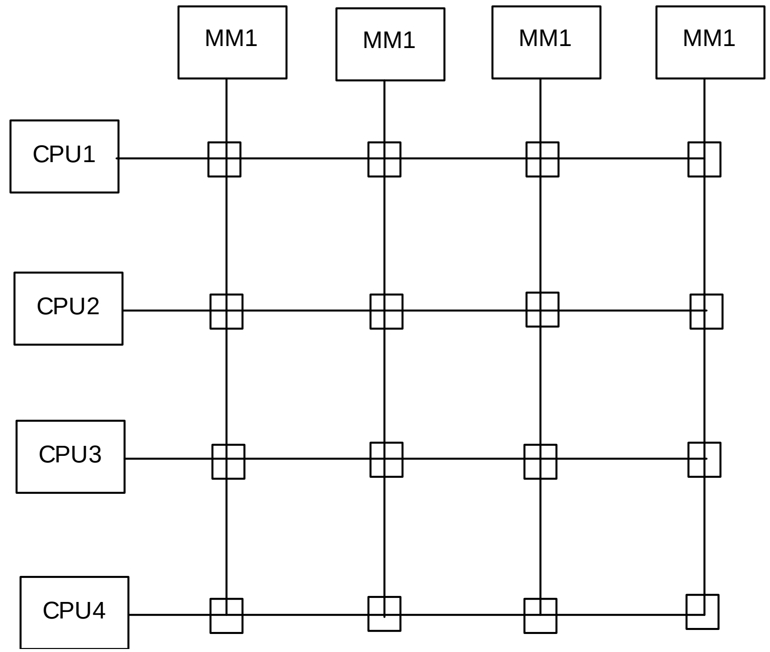
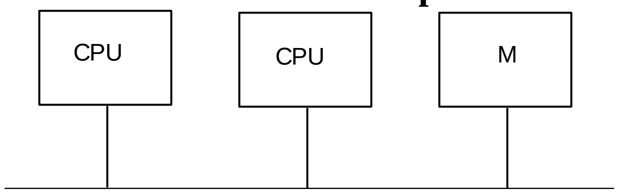


Fig: crossbar switch

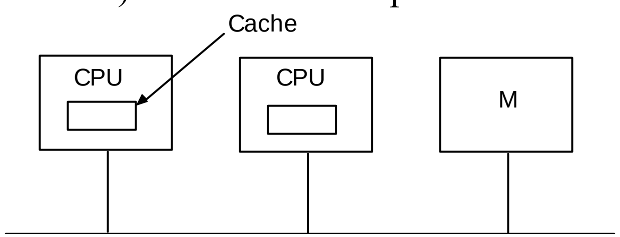
The crossbar switch organization consists of no of cross points that are placed at the interconnection between the processor bus and memory module both. Fig shown crossbar interconnection between four CPU and four memory modules.

The small square in each cross point is a switch that determine the path form processor to memory modules. Each switch point has control logic to set up the transfer path between the processor and memory. It examine the address that is placed in the bus to determine. Whether its particular modules is being address. Its also restore multiple request for access to the same memory modules on predetermine priority basic. A crossbar switch organization support simultaneous transfer form all memory modules because there is a separated path associated with each module.

Cache coherence and MESI protocol:



a) Bus based multiprocessors without cache.



b) Busbased multiprocessor with cache.

The simplest multiprocessor based on single bus to or more CPU and one or more memory modules all used the same bus for communication. When a CPU want to read a memory wrod , it first checked to see if the bus is busy, if the bus is ideal the CPU put the address of the word it wants on the bus asserts a few control signal and waits until the memory puts a desire words on the bus.

If the bus is busy when the CPU wants to read/write memory, the CPU just wait until the bus becomes ideal with 2 or 3 CPU contention for bus will be manageable with 32 or 64 bit will be unbearable. Most of the CPU will be ideal most of this time.

The solution to these problem is to add a cache to each CPU since many reads can now satisfy out of local cache. There will be much less bus traffic and system can support more CPU.

When a processor find the word in cache during read operation the main memory is not involves in the transfer. If the operation is to write there are two commonly used procedure to update memory in write through method. Both cache and main memory are updated with every write operation.

In the write policy only the cache is updated and location is marked so that it can be copied latter into the main memory.

To ensure the ability of the system to executed the memory operation correctly multiple copies must be identical. These requirement cache coherence problem read only data can safely be replicated without cache coherence to illustrated the problem consider three processor configuration with private cache shown in fig.

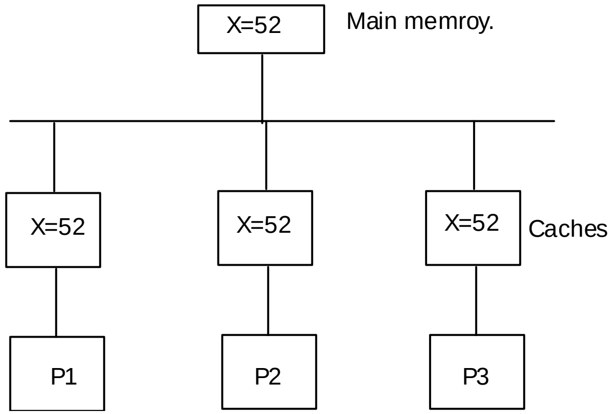
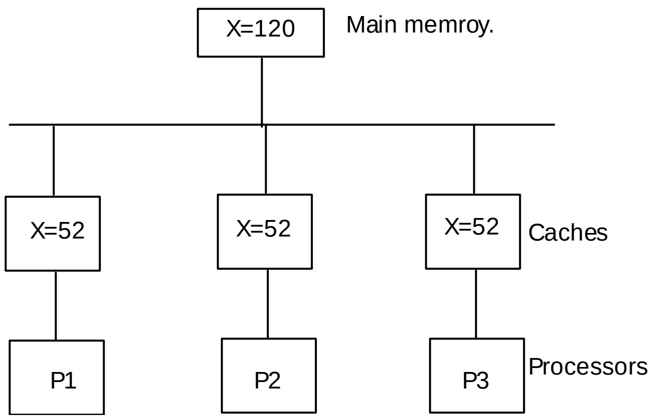
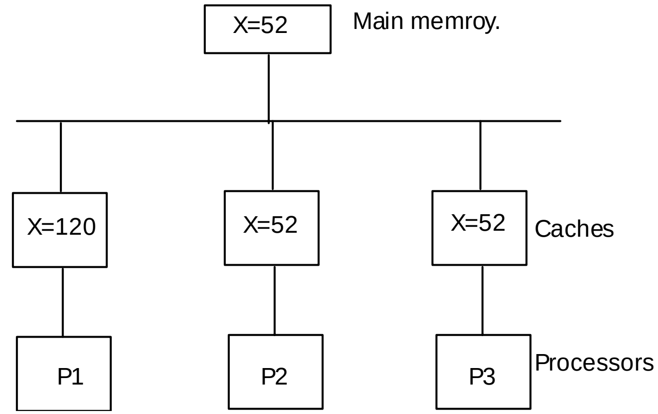


Fig: cache configuration after load on x.

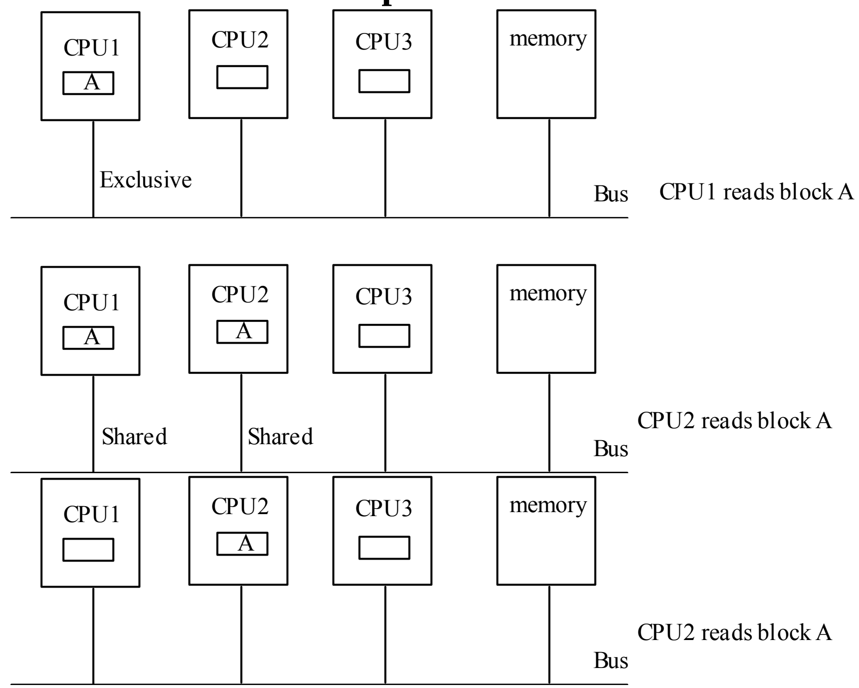


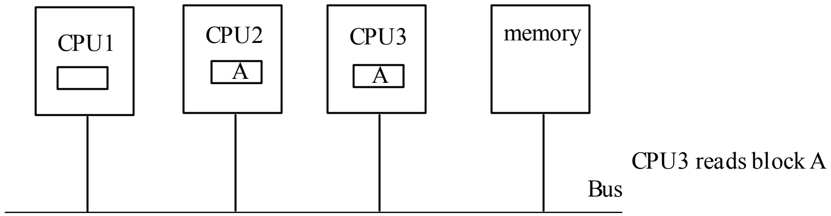
a) with write through cache policy.



b) with write back cache policy.

MESI cache coherence protocol:





To provide cache consistency, cache after supports a protocol known as MESI(Modified Exclusive shared invalid) each cache entry can be is one of the following four steps.

1. Invalid: The cache entry doesn't contain valid data.
2. Shared: Multiple Caches may hold the line.
3. Exclusive: No other cache hold the line.
4. Modified: The entry is valid. Memory is invalid.

The first time the memory is read, the line referenced is fetched into the cache of CPU reading memory and marked as being in 'E' state. Since it is the only copy in a cache as shown in fig 'a'. Another CPU may also fetch the same line and cache it both copies are marked as being in 's' state as shown in fig b. If CPU two writes to the cache line it is holding in 's' state it puts out invalidate signal on the bus telling all other CPUs to discard their copies. The copy catch now goes to M state as shown in fig 'c'. If CPUs reads the line, cpu2 which now owns the line knows that copy in memory is not valid so it asserts the signal on the bus telling CPUs to please wait which it writes its line back to the memory when it is finished. CPUs fetches a copy and the line is marked as shared in both cases as shown in fig 'd'.

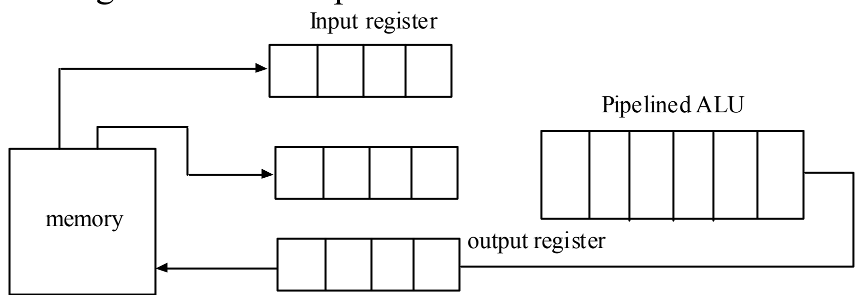
Vector Computer:-

Consider two vectors (one dimensional array) of numbers A and B. We would like to add there and place the result in c. In the example.

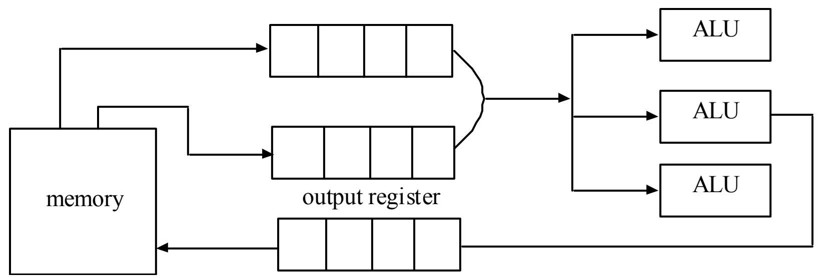
$$\begin{bmatrix} 1.5 \\ 7.1 \\ 6.9 \\ 100.5 \\ 0 \\ 59.7 \end{bmatrix} + \begin{bmatrix} 2.0 \\ 39.7 \\ 1000.003 \\ 11 \\ 21.1 \\ 19.7 \end{bmatrix} = \begin{bmatrix} 3.5 \\ 46.8 \\ 1006.903 \\ 111.5 \\ 21.1 \\ 79.4 \end{bmatrix}$$

$$A + B = C$$

This requires six separate addition. We can seed up this by introducing some form of parallelism.



b) Pipelined ALU



c) Parallel ALU

Floating point operations are complex. There is opportunity for decomposing floating point operation into stages so that different

stages can operate on different sets of data concurrently. Floating point addition is broken up into four stages: Compare, shift, Add and Normalize. A vector of numbers is presented sequentially to the first stage as the processing proceeds four different stets of numbers will be operated on concurrently in the pipeline.

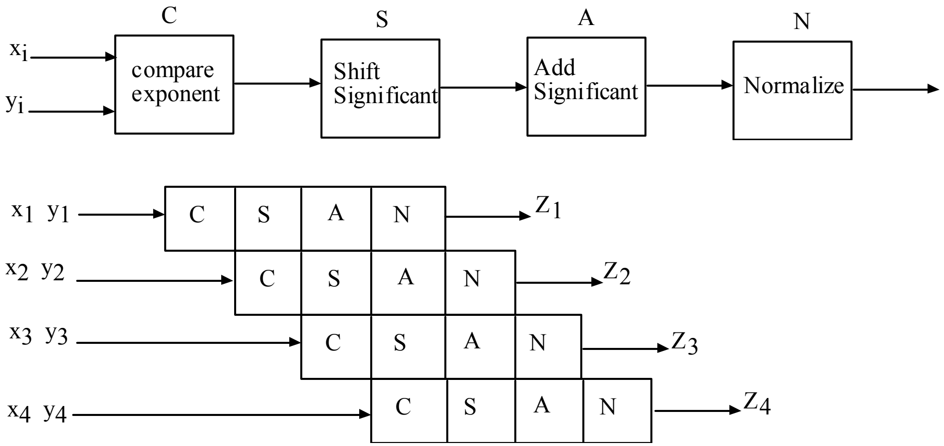


Fig: pipelined ALU